Git “under the hood”

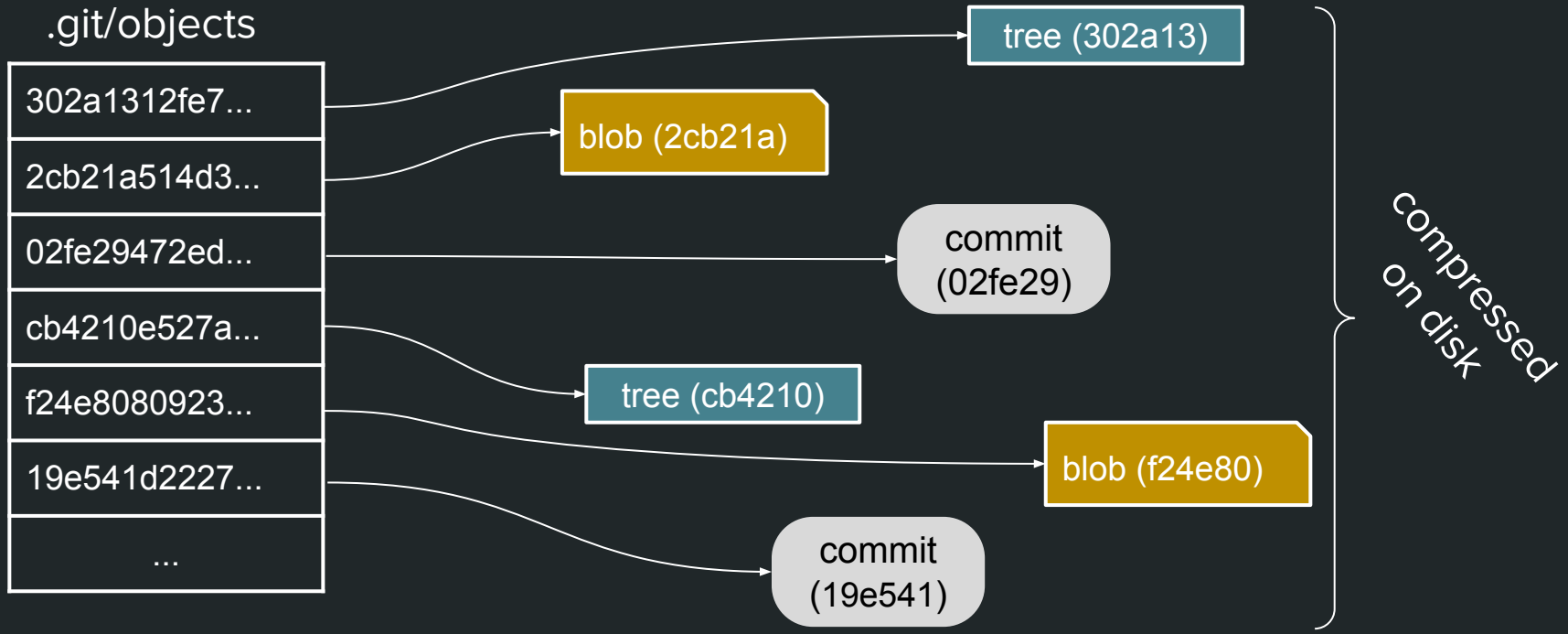
Matheus Tavares Bernardino
https://matheustavares.gitlab.io

Based on ***The Zen of Git***, by Tianyu Pu
https://speakerdeck.com/tianyupu/the-zen-of-git

https://xkcd.com/1597/

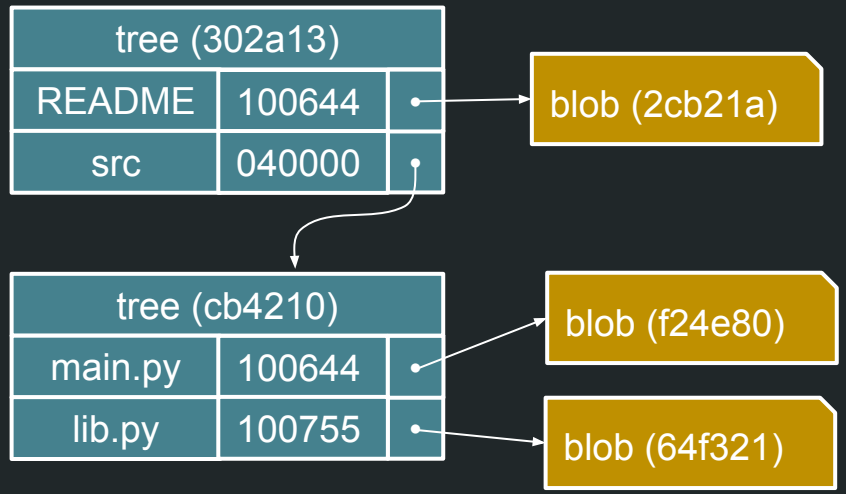
How Git works?

A HashSet


.git/objects
302a1312fe7...
2cb21a514d3...
02fe29472ed...
cb4210e527a...
f24e8080923...
19e541d2227...
...
tree (302a13)
blob (2cb21a)
commit (02fe29)
tree (cb4210)
blob (f24e80)
commit (19e541)
compressed on disk

Objects' interactions

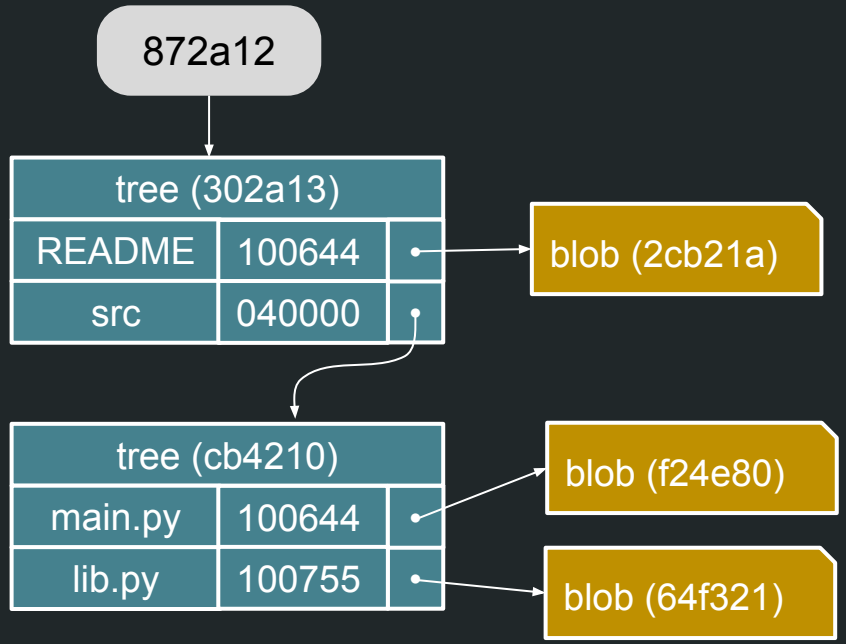
```
$ tree repo
repo
├── README
└── src
    ├── main.py
    └── lib.py
```



Objects' interactions

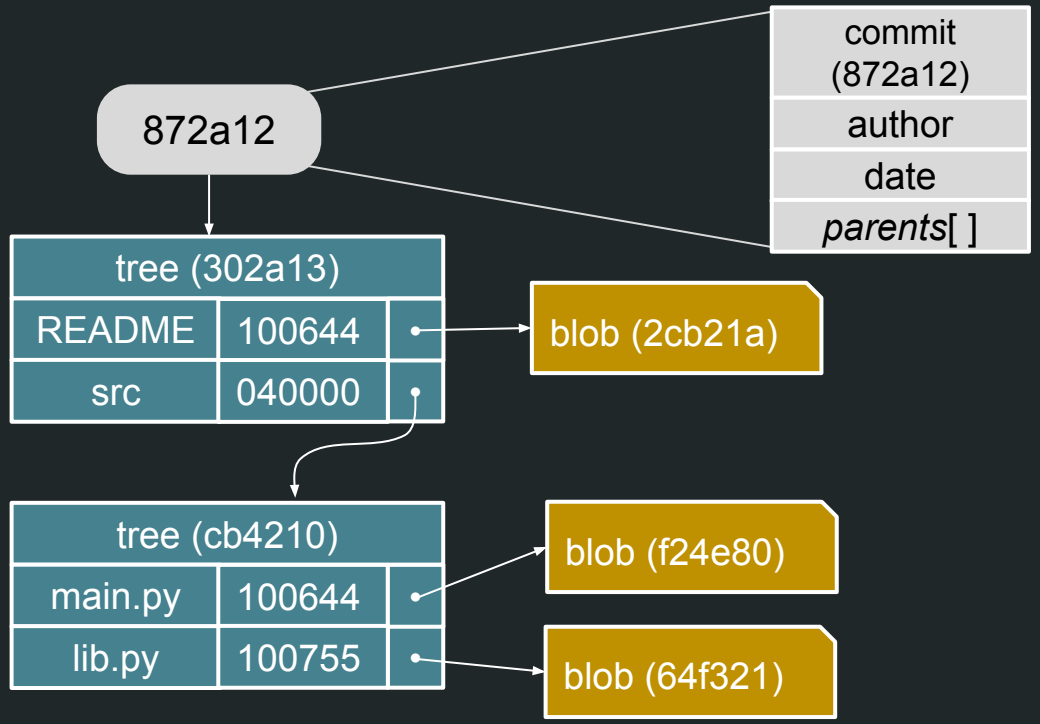
A *commit* stores a snapshot of the project

(a *tree*, not a *diff!*)



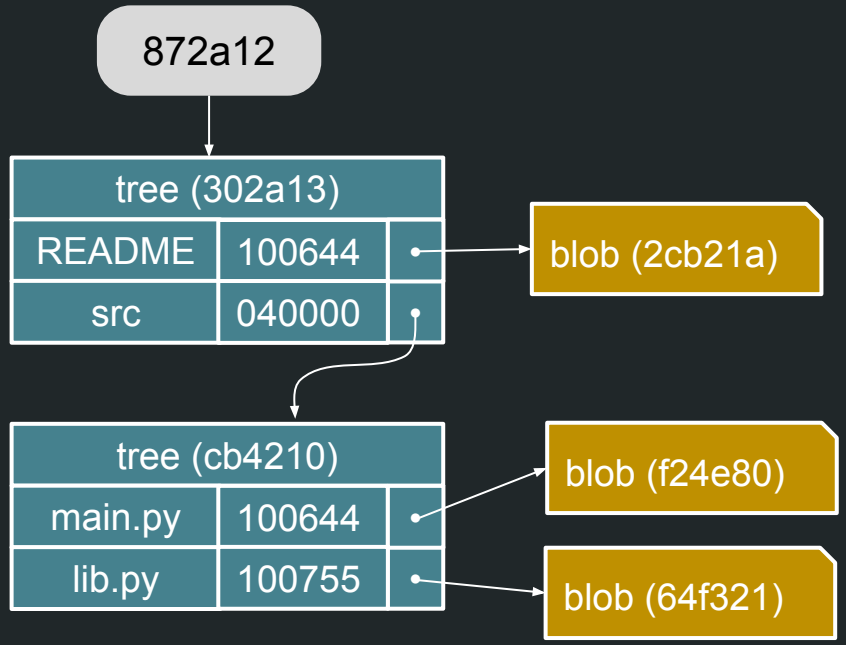
Objects' interactions

Commits also contain metadata.



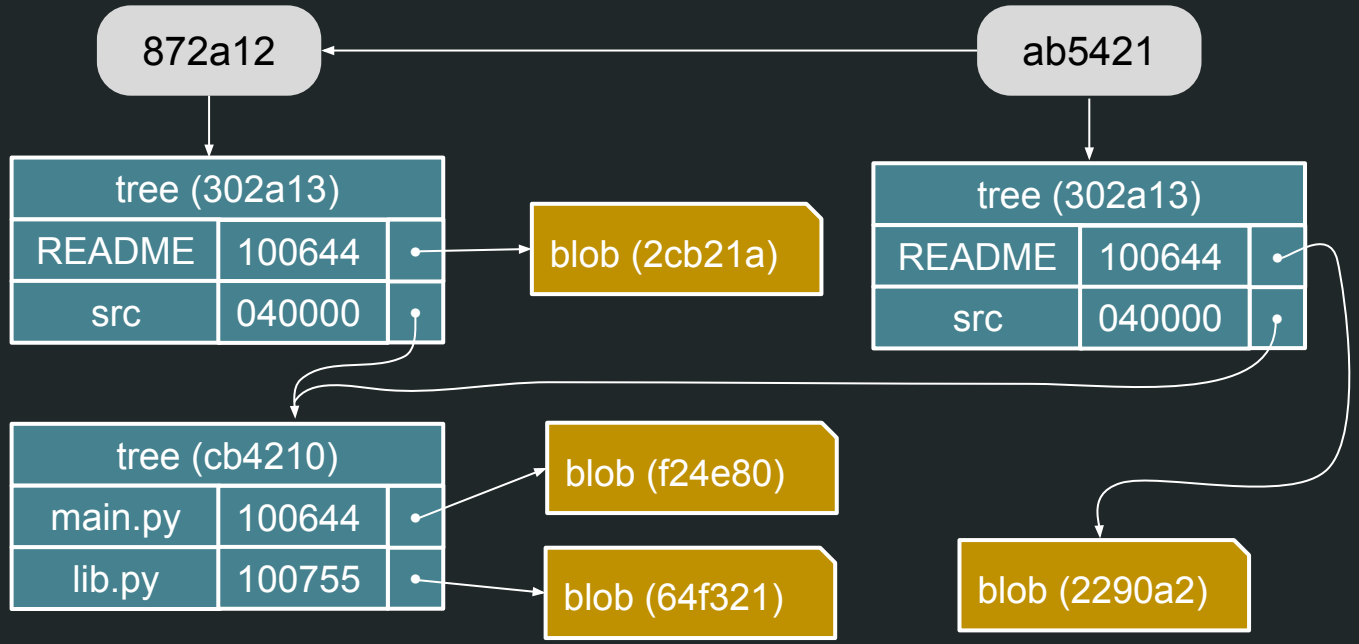
Objects' interactions

$ vim README
$ git commit



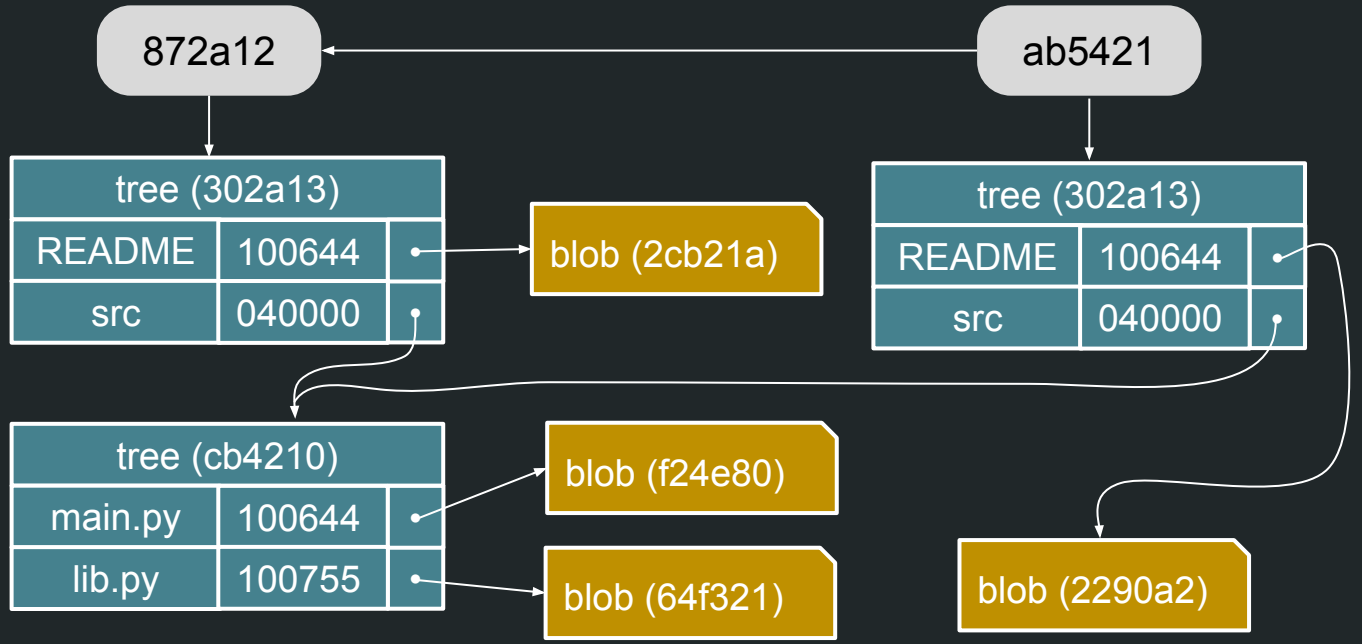
Objects' interactions

$ vim README
$ git commit



Directed acyclic graph

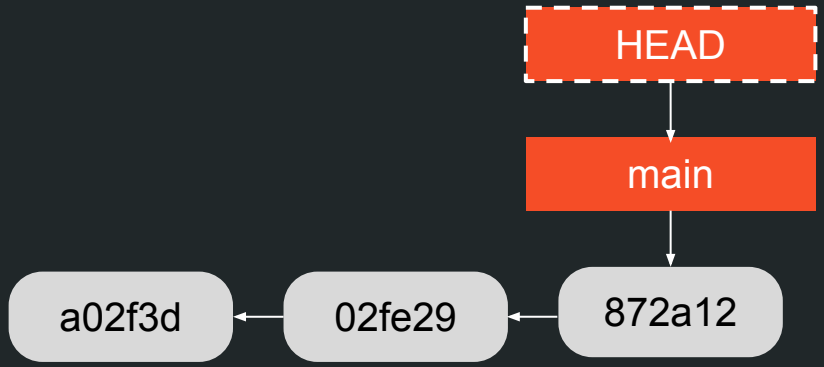
$ vim README
$ git commit



References

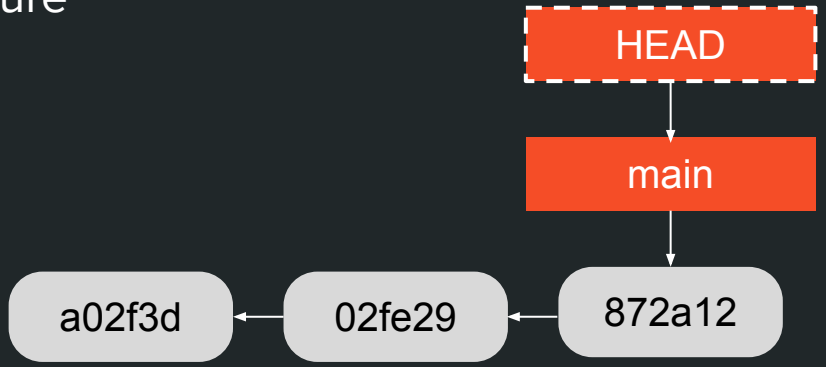
References

- branches
- tags
- HEAD



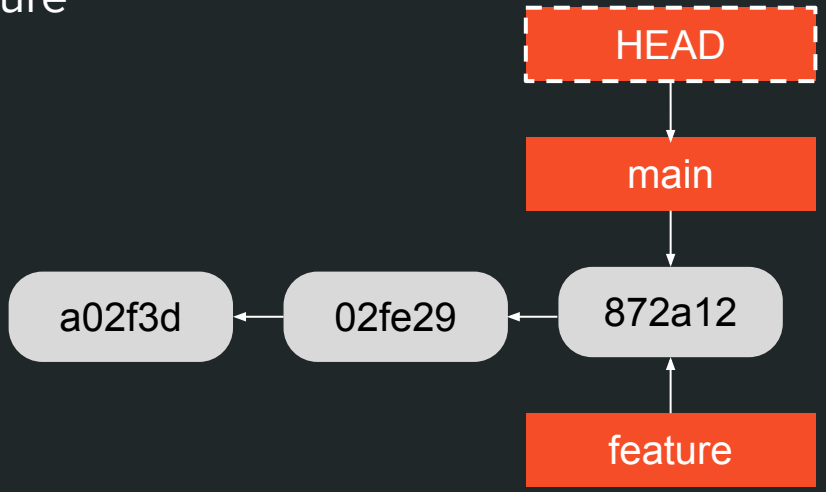
Refs: branches

$ git branch feature



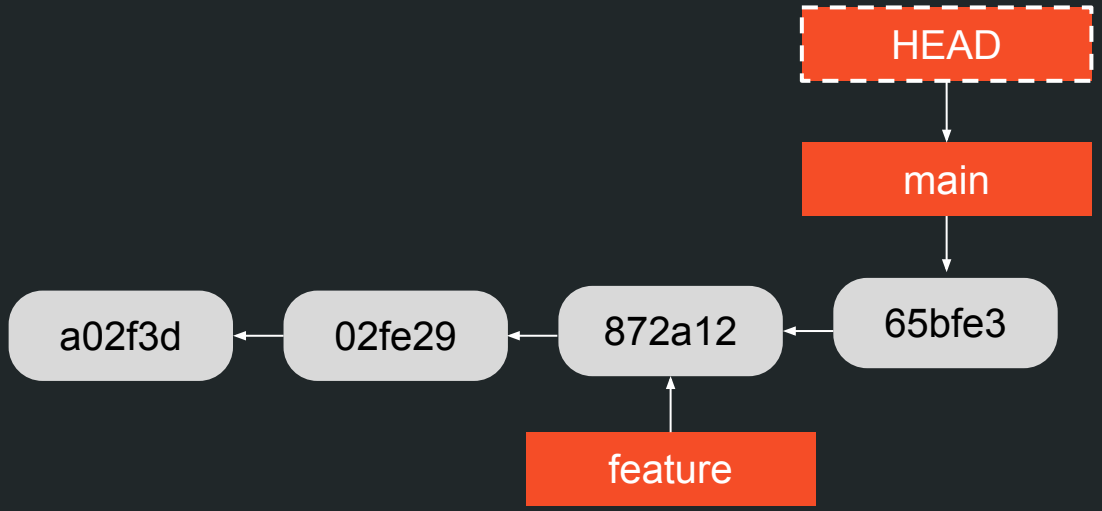
Refs: branches

$ git branch feature



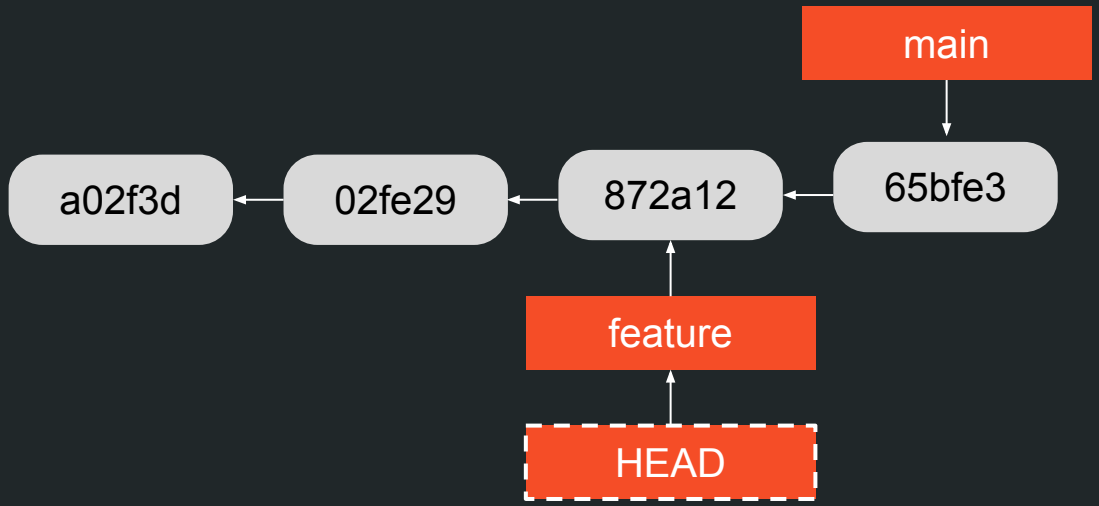
Refs: branches

$ git commit



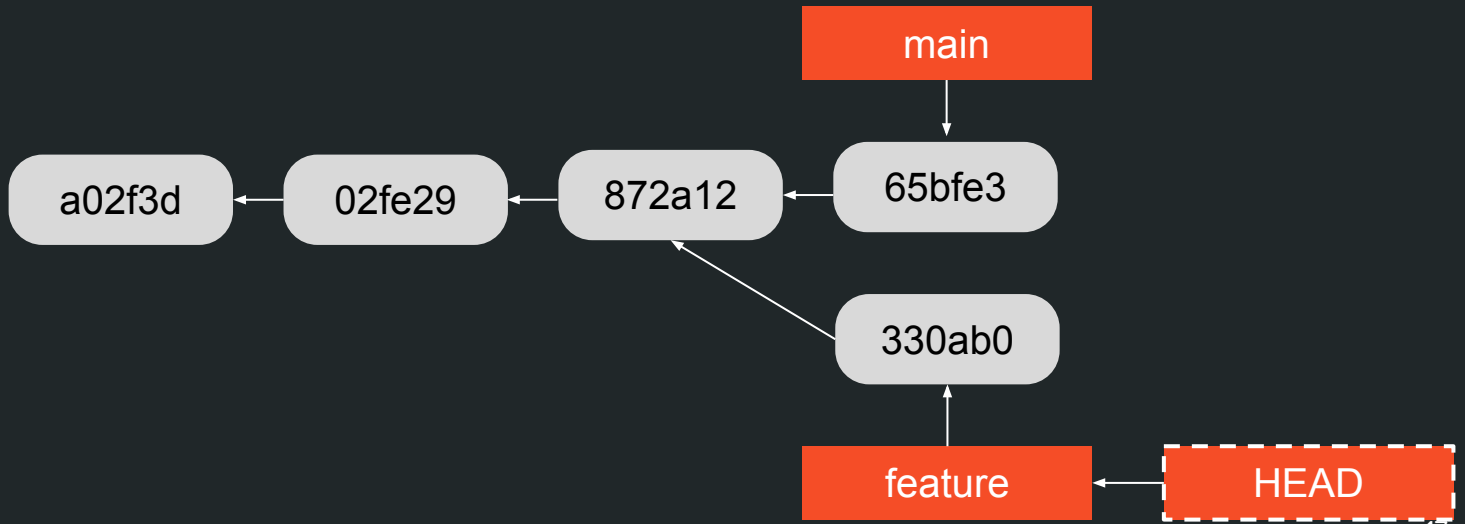
Refs: branches

$ git checkout feature



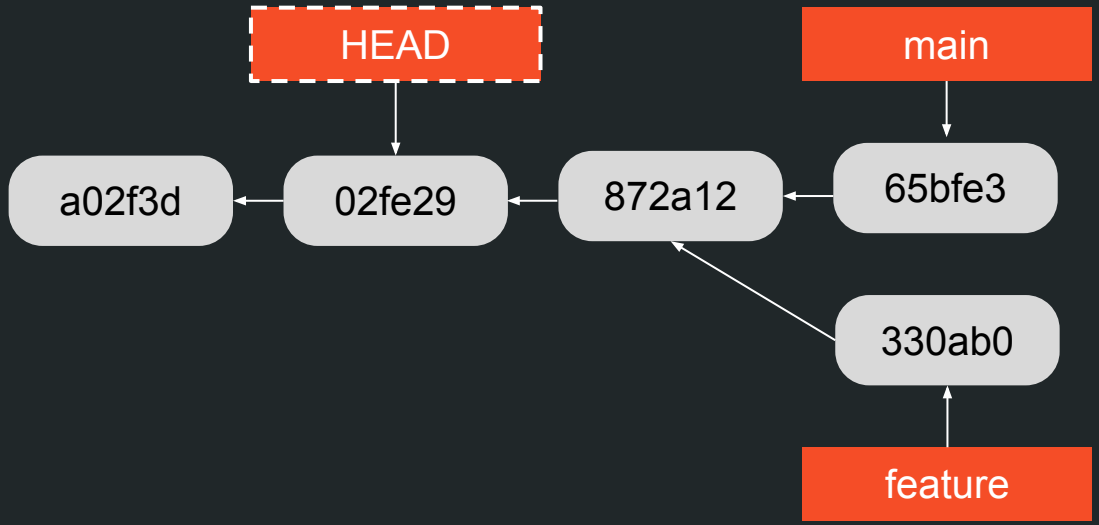
Refs: branches

$ git commit



Refs: branches

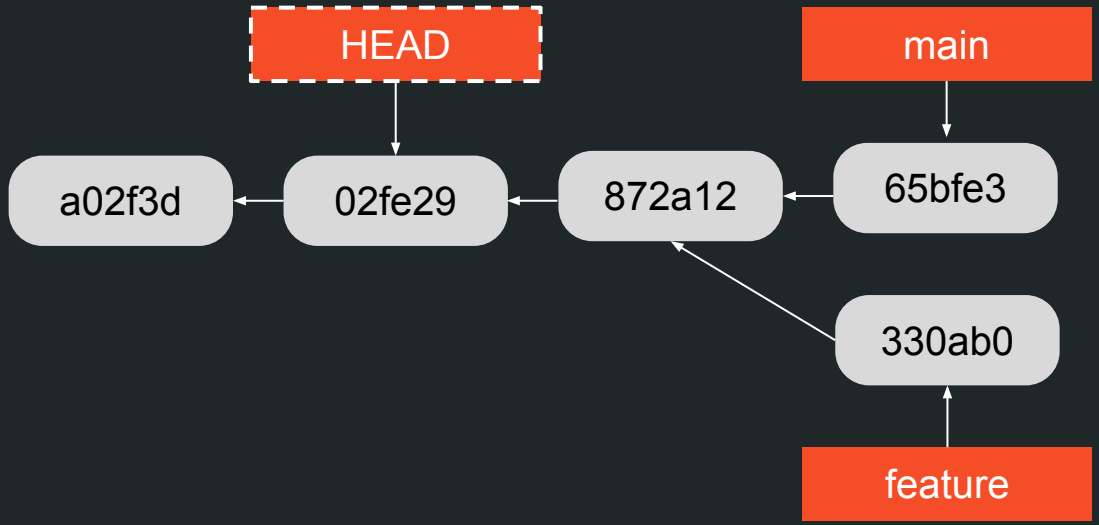
$ git checkout 02fe29



Refs: branches

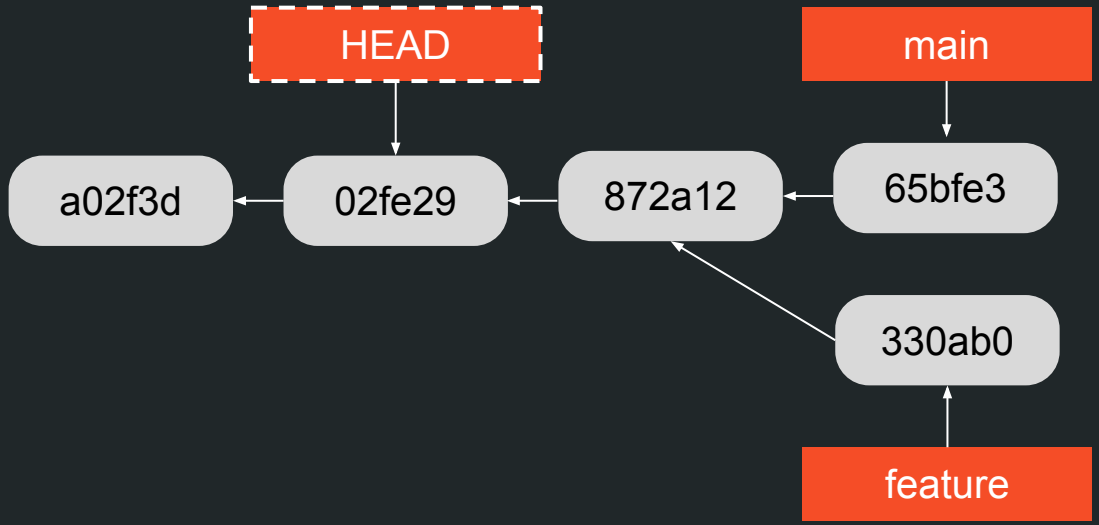
$ git checkout 02fe29

You are in “detached HEAD” state.



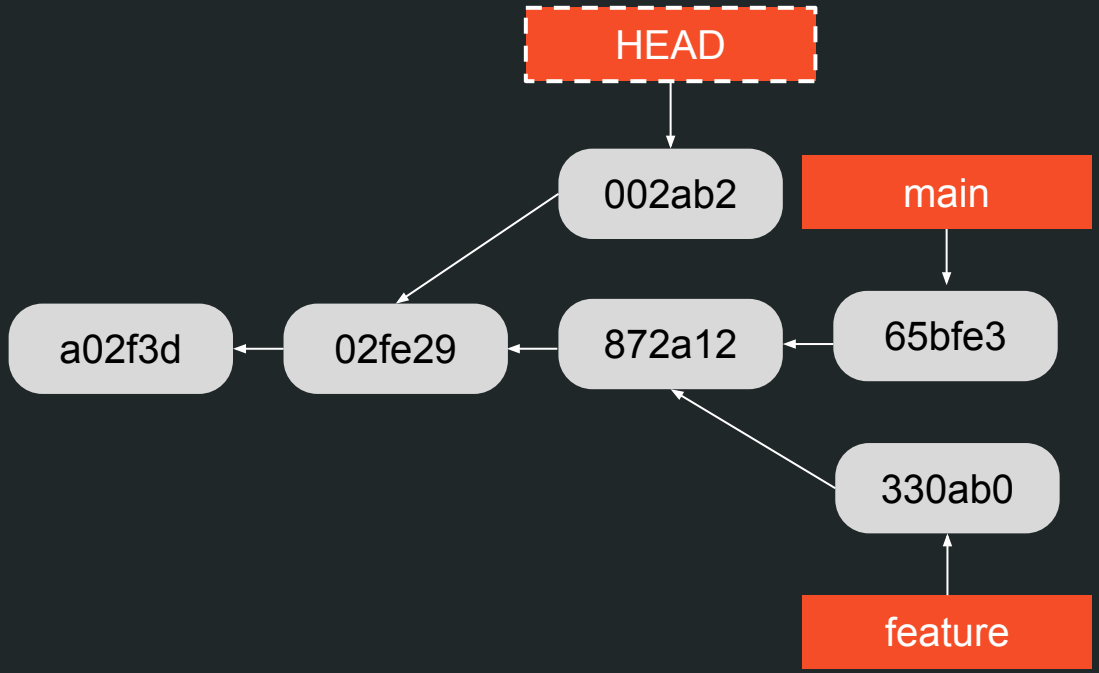
Refs: branches

$ git commit



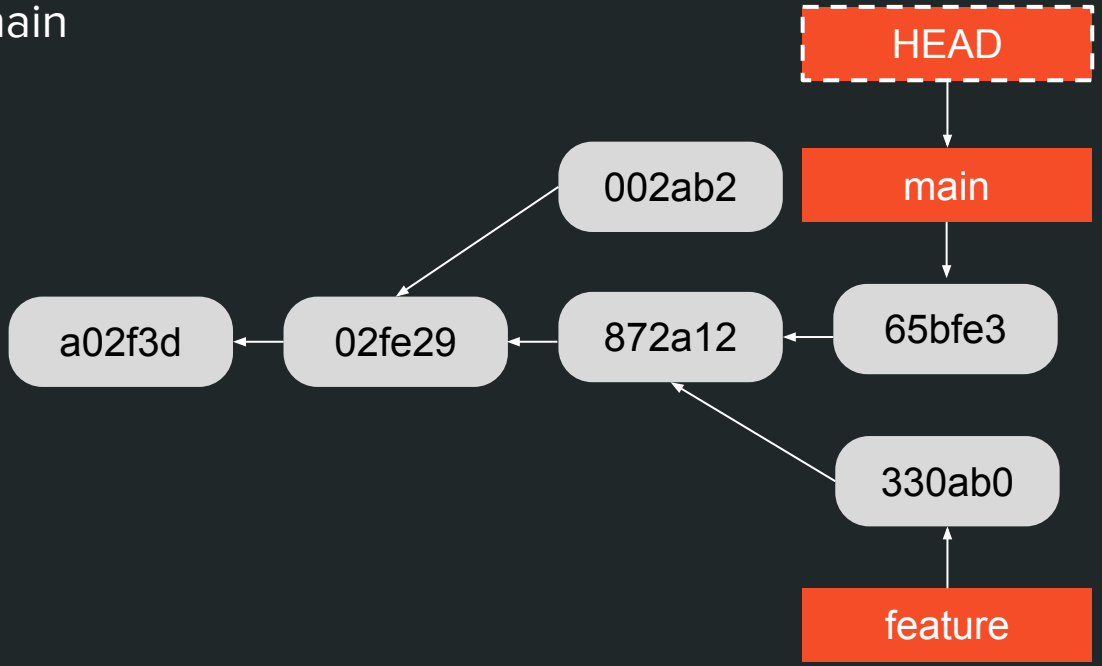
Refs: branches

$ git commit



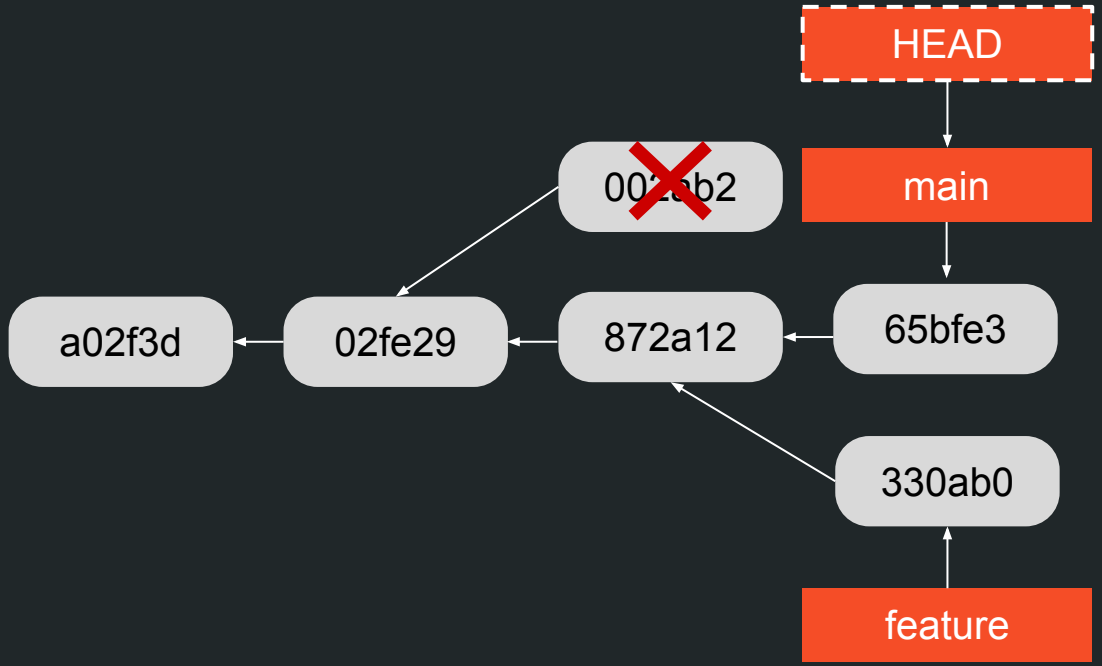
Refs: branches

$ git checkout main



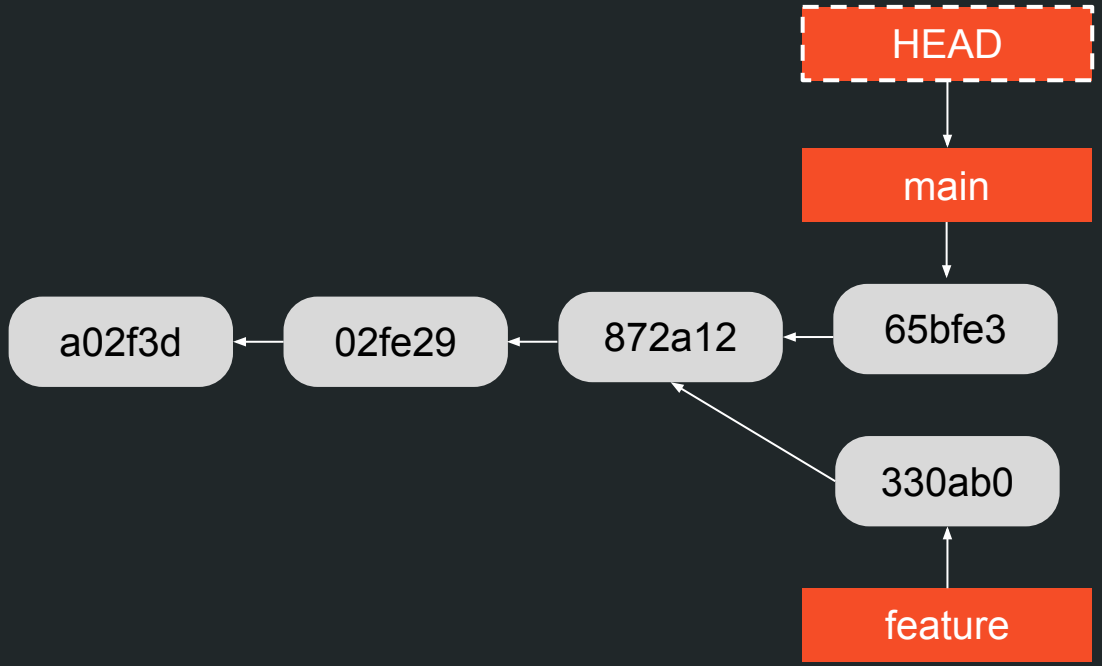
Refs: branches

$ git gc --auto



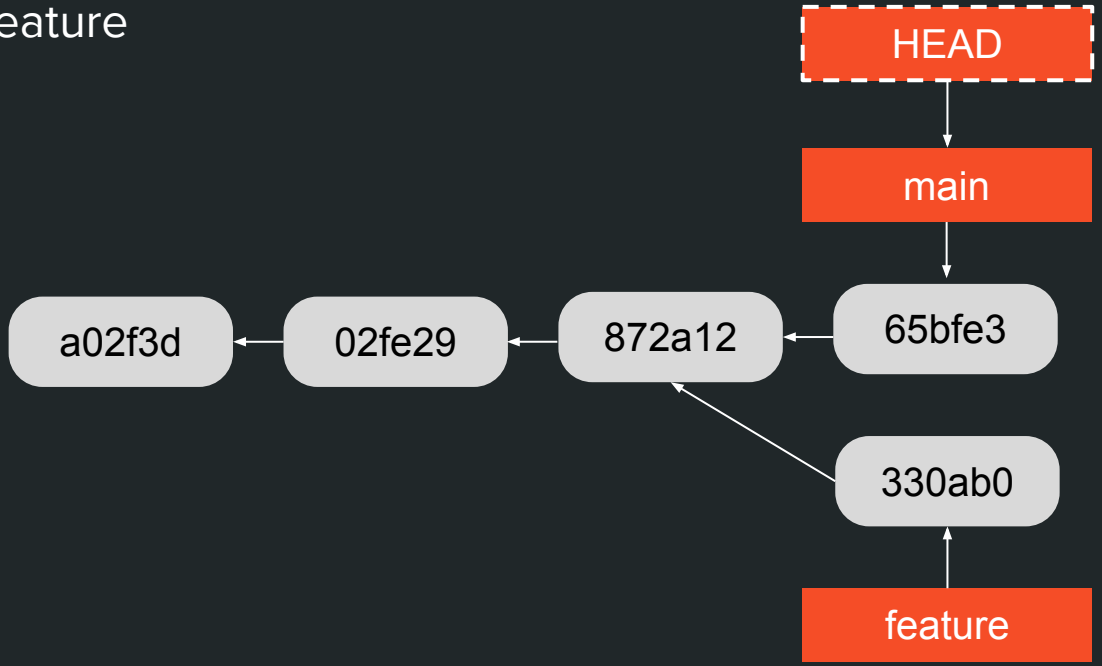
Refs: branches

$ git gc --auto



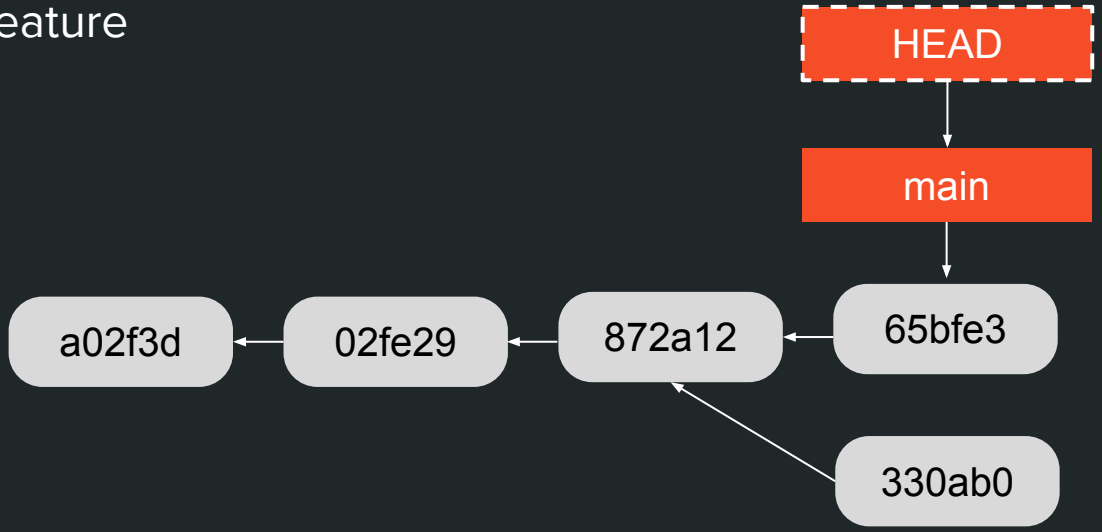
Refs: branches

$ git branch -D feature



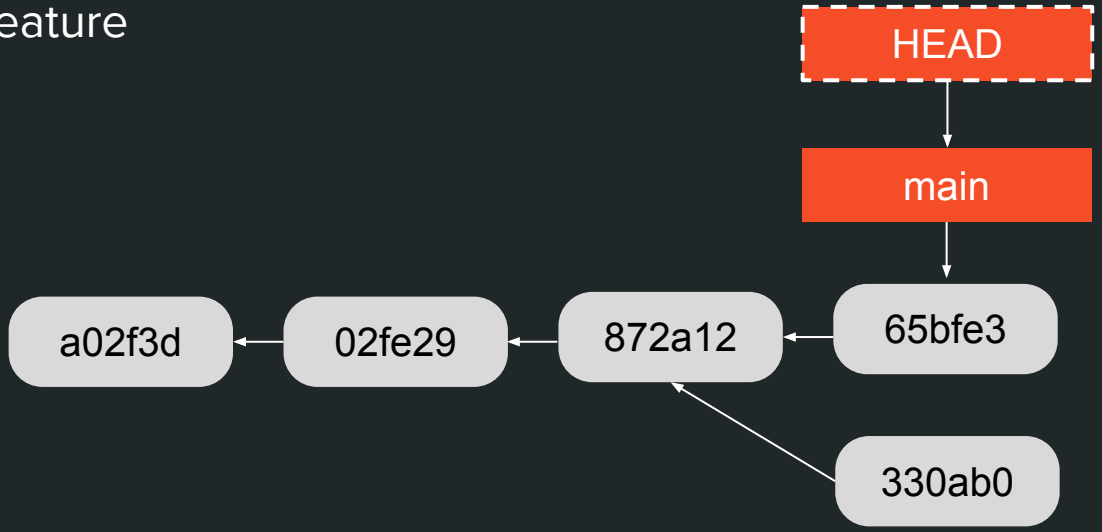
Refs: branches

$ git branch -D feature



Refs: branches

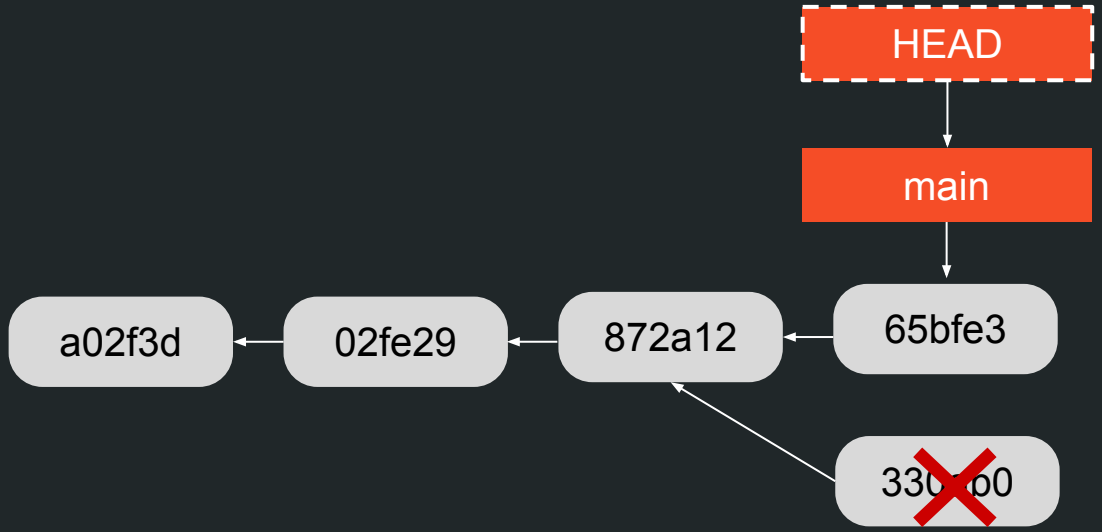
$ git branch -D feature



Still recoverable!
See *git reflog*

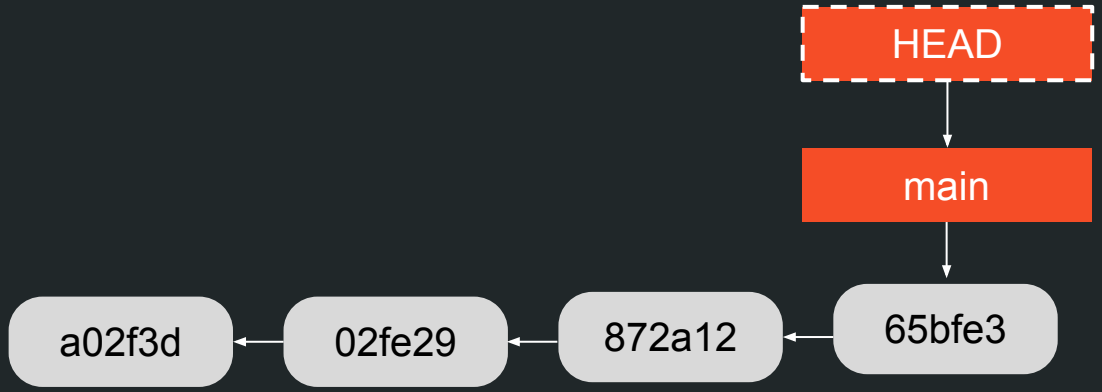
Refs: branches

$ git gc --auto



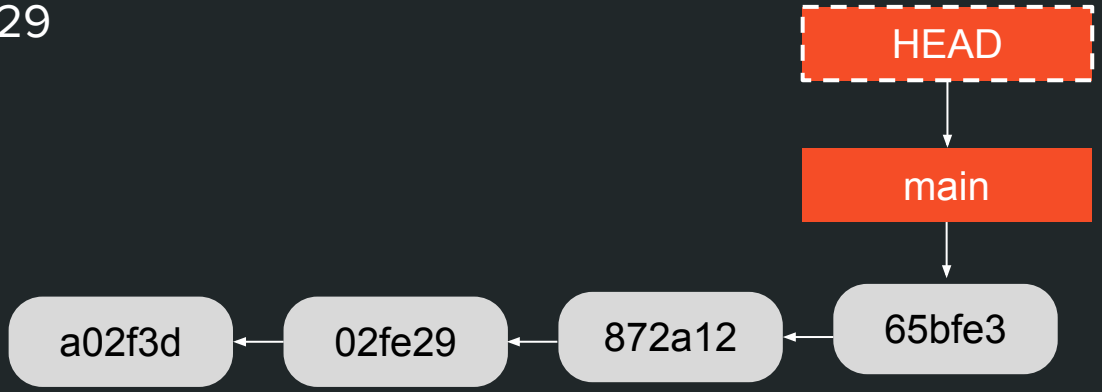
Refs: branches

$ git gc --auto



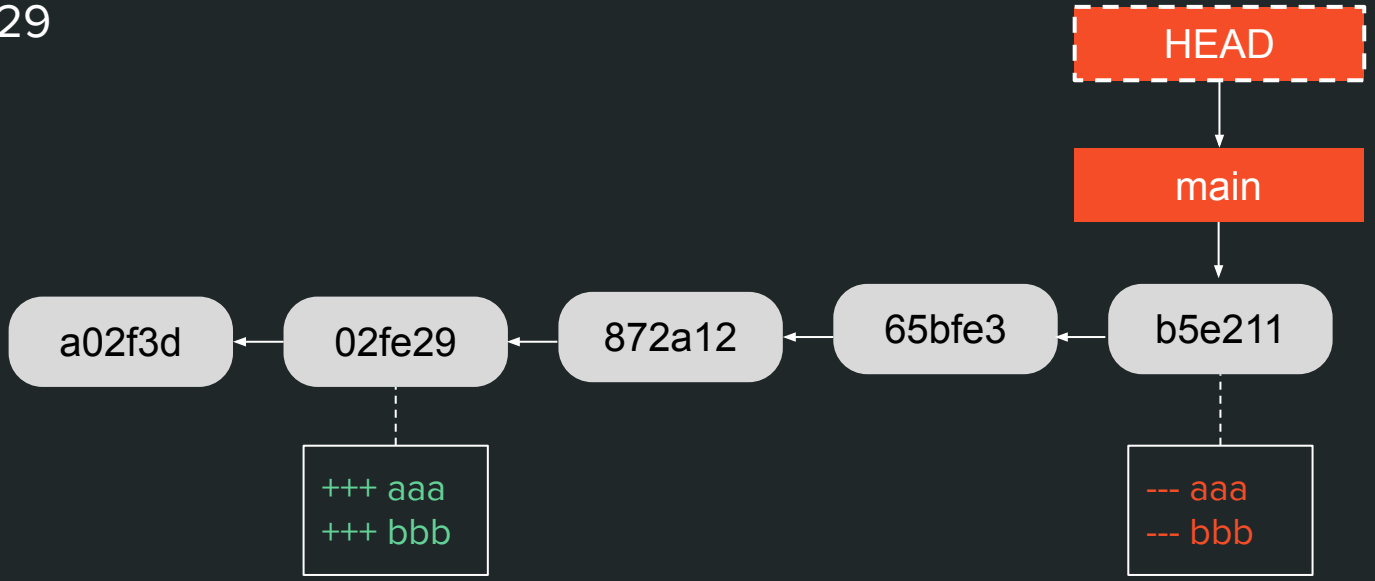
Refs: branches

$ git revert 02fe29



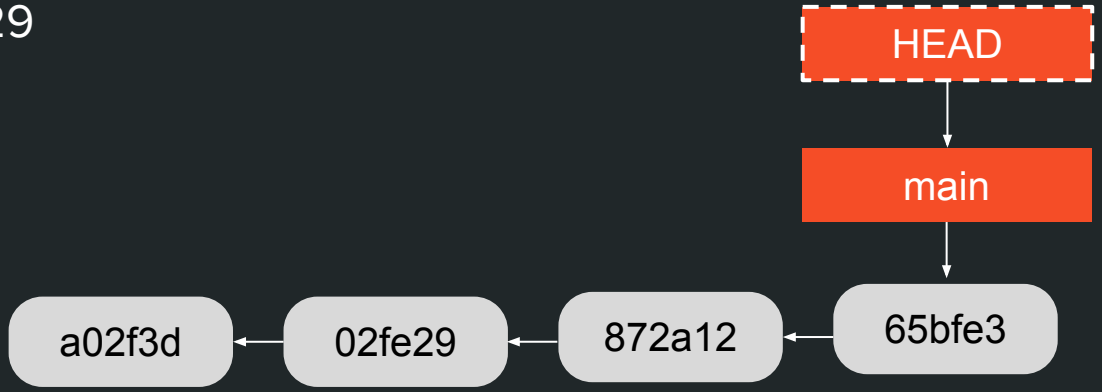
Refs: branches

$ git revert 02fe29



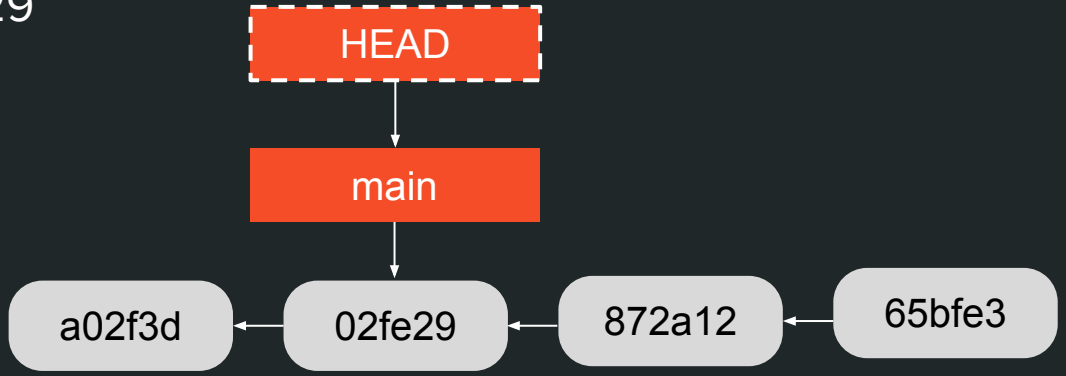
Refs: branches

$ git reset 02fe29

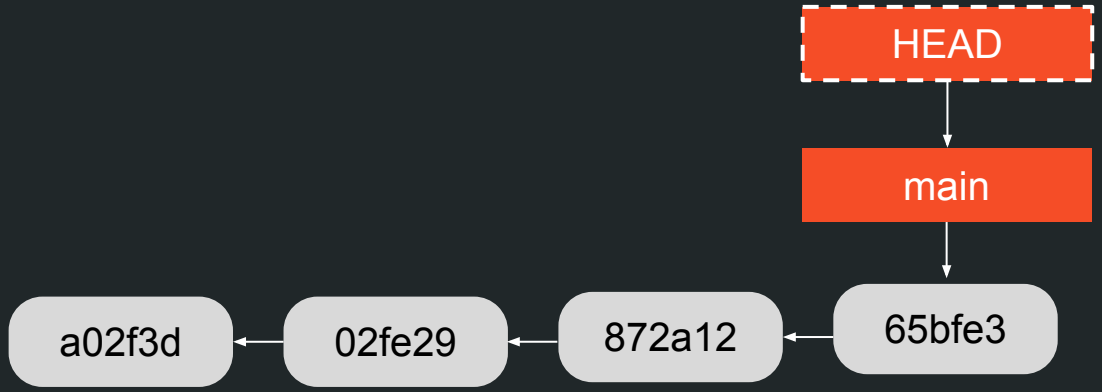


Refs: branches

$ git reset 02fe29

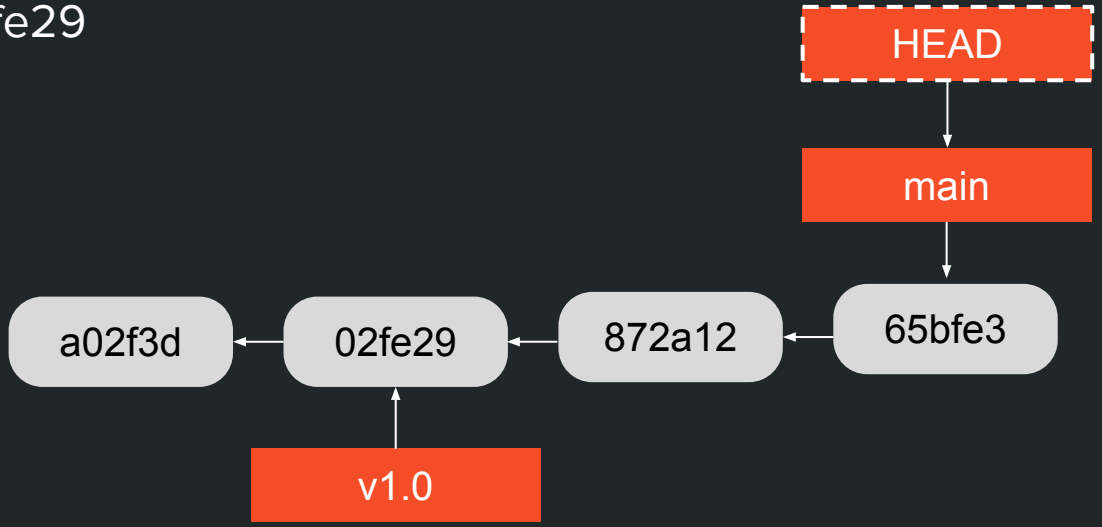


Refs: tags



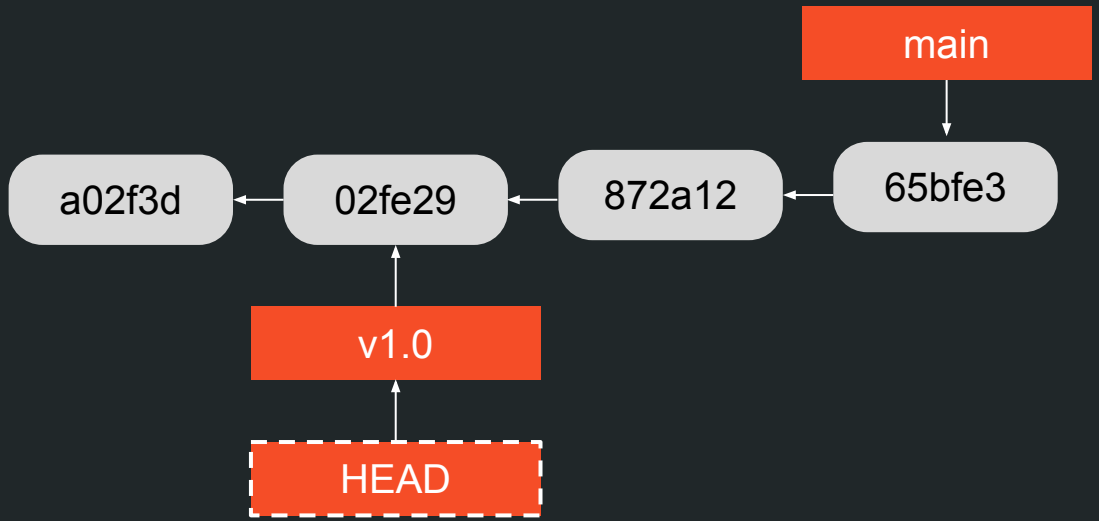
Refs: tags

$ git tag v1.0 02fe29



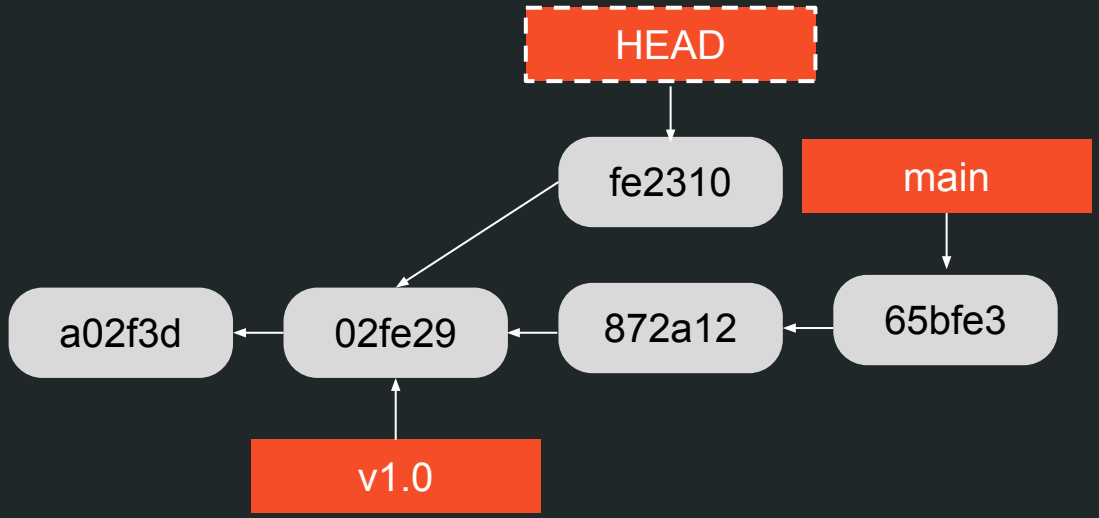
Refs: tags

$ git checkout v1.0



Refs: tags

$ git commit

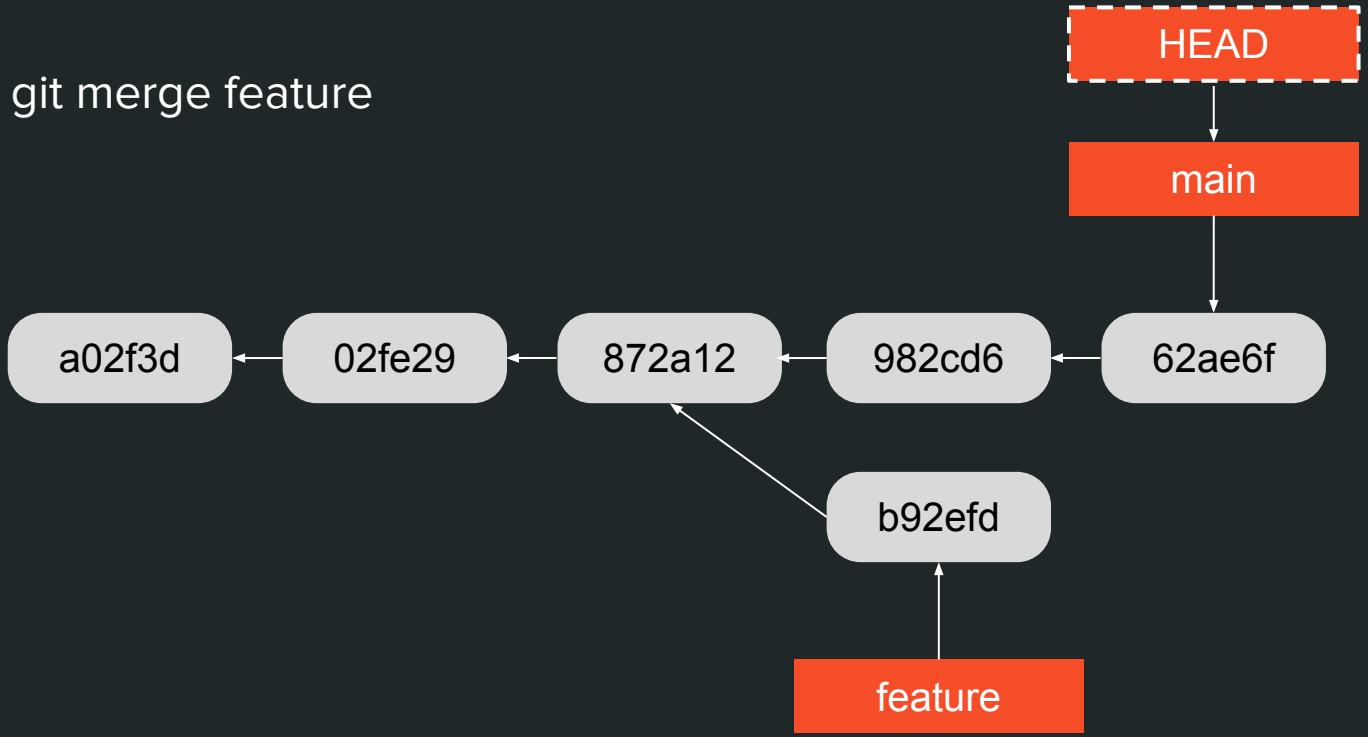


Merging

https://giphy.com/gifs/git-merge-cFkiFMDg3iFol

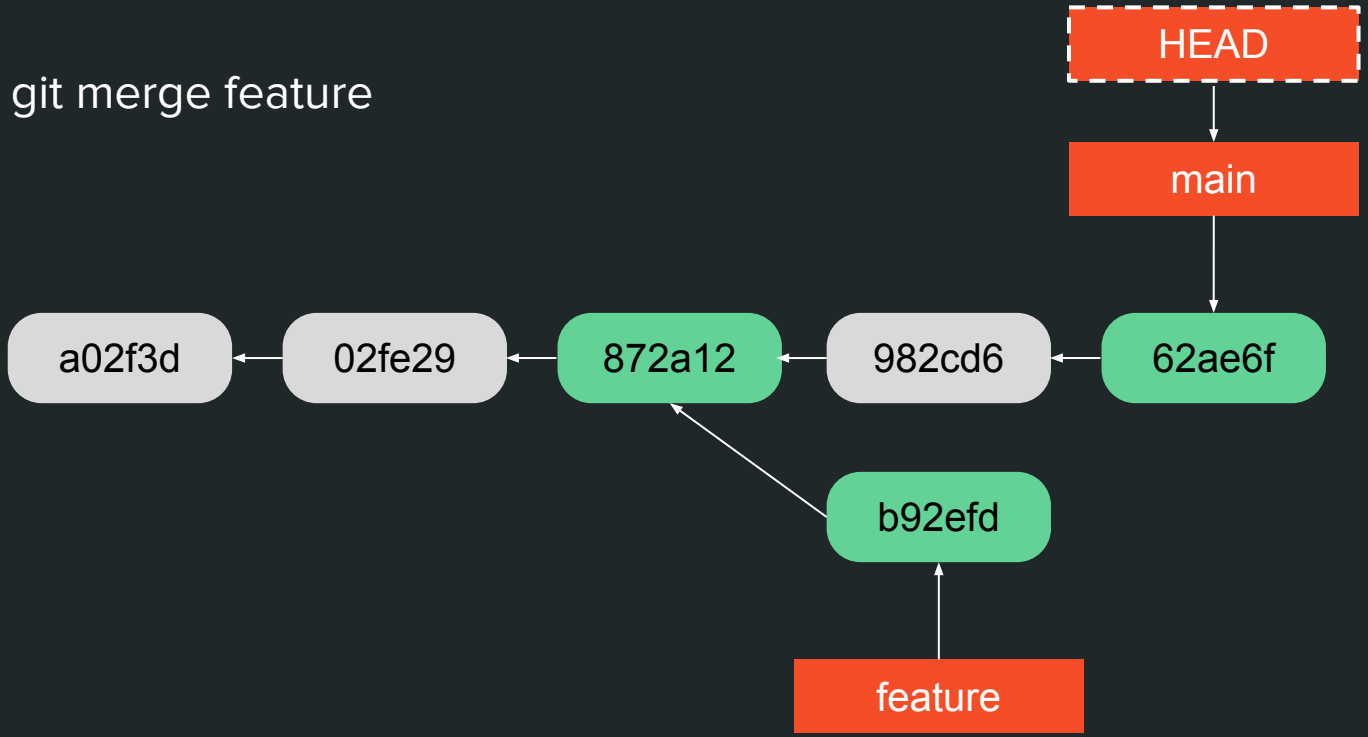
Merging

$ git merge feature



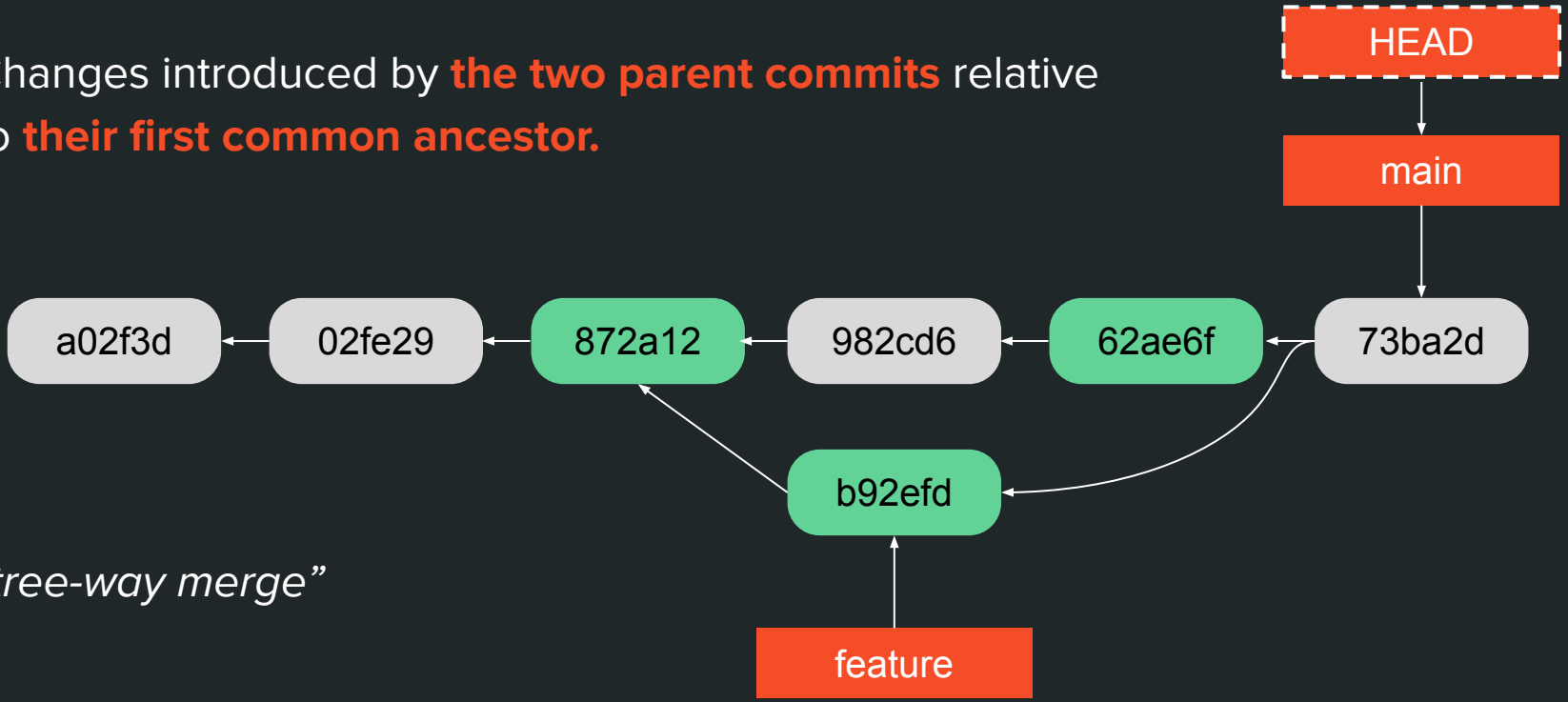
Merging

$ git merge feature



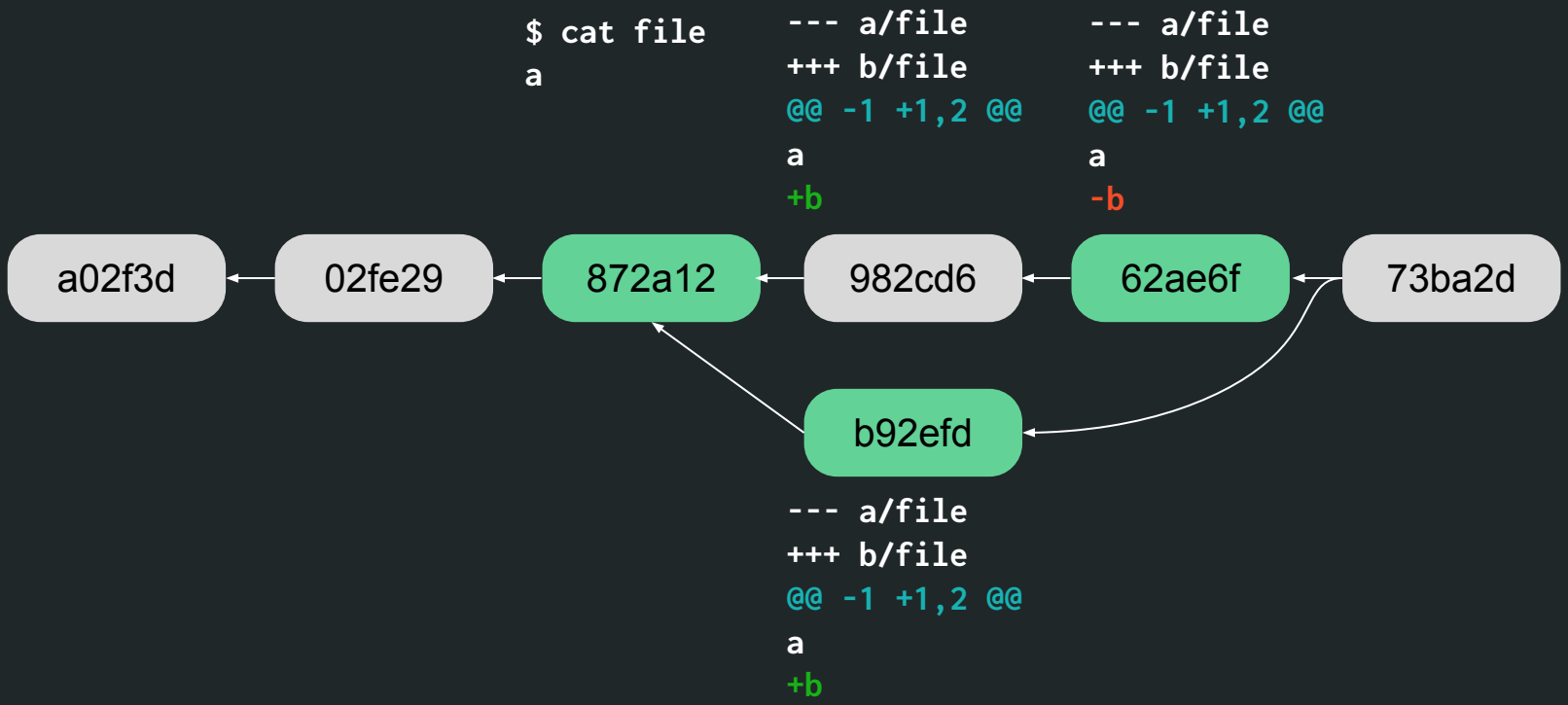
Merging

Changes introduced by **the two parent commits** relative to **their first common ancestor.**

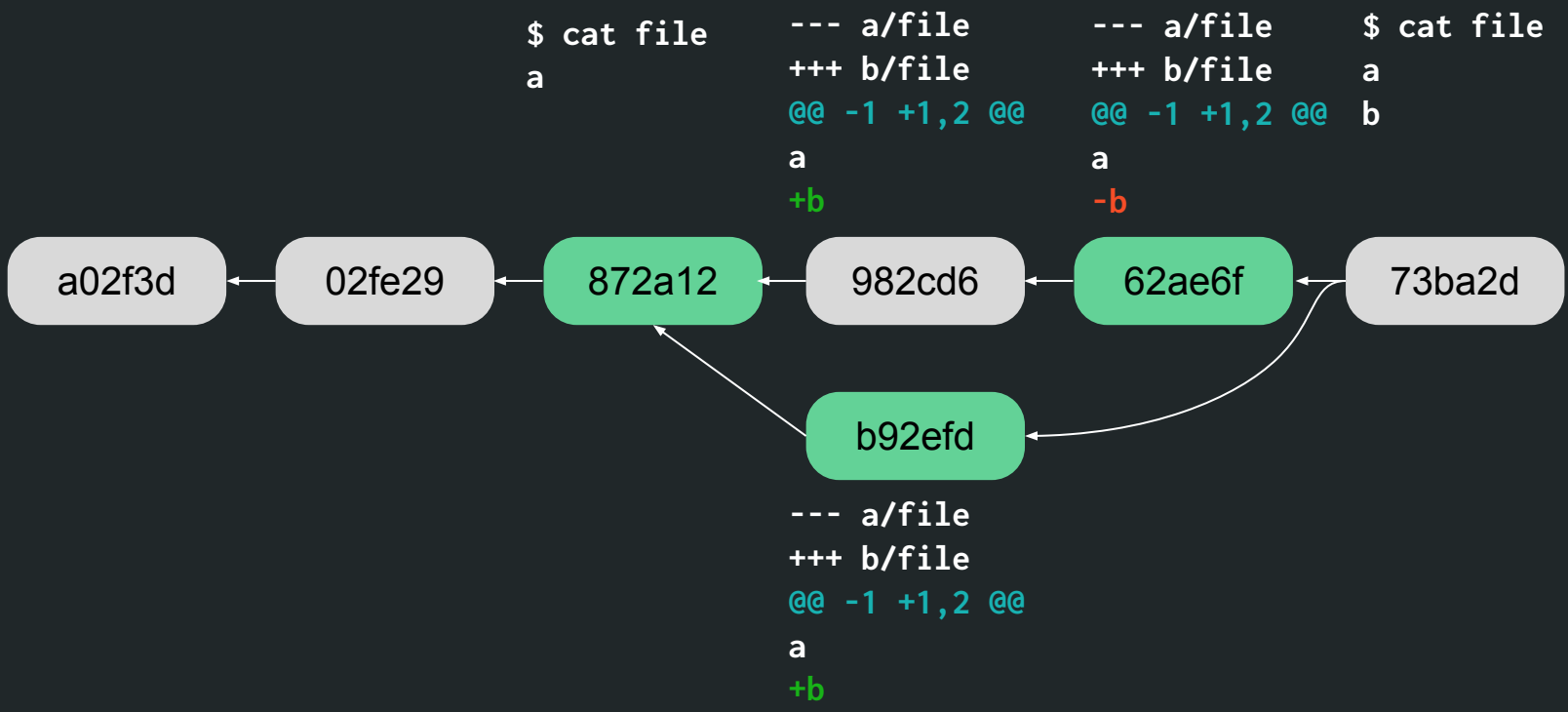


“tree-way merge”

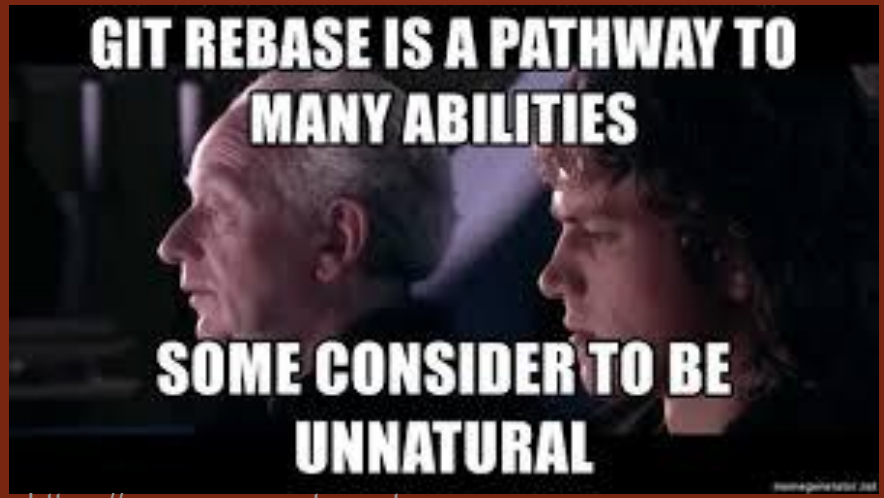
Merging


$ cat file
a
--- a/file
+++ b/file
@@ -1 +1,2 @@
a
+b
--- a/file
+++ b/file
@@ -1 +1,2 @@
a
-b
a02f3d
02fe29
872a12
982cd6
62ae6f
73ba2d
b92efd
--- a/file
+++ b/file
@@ -1 +1,2 @@
a
+b

Merging

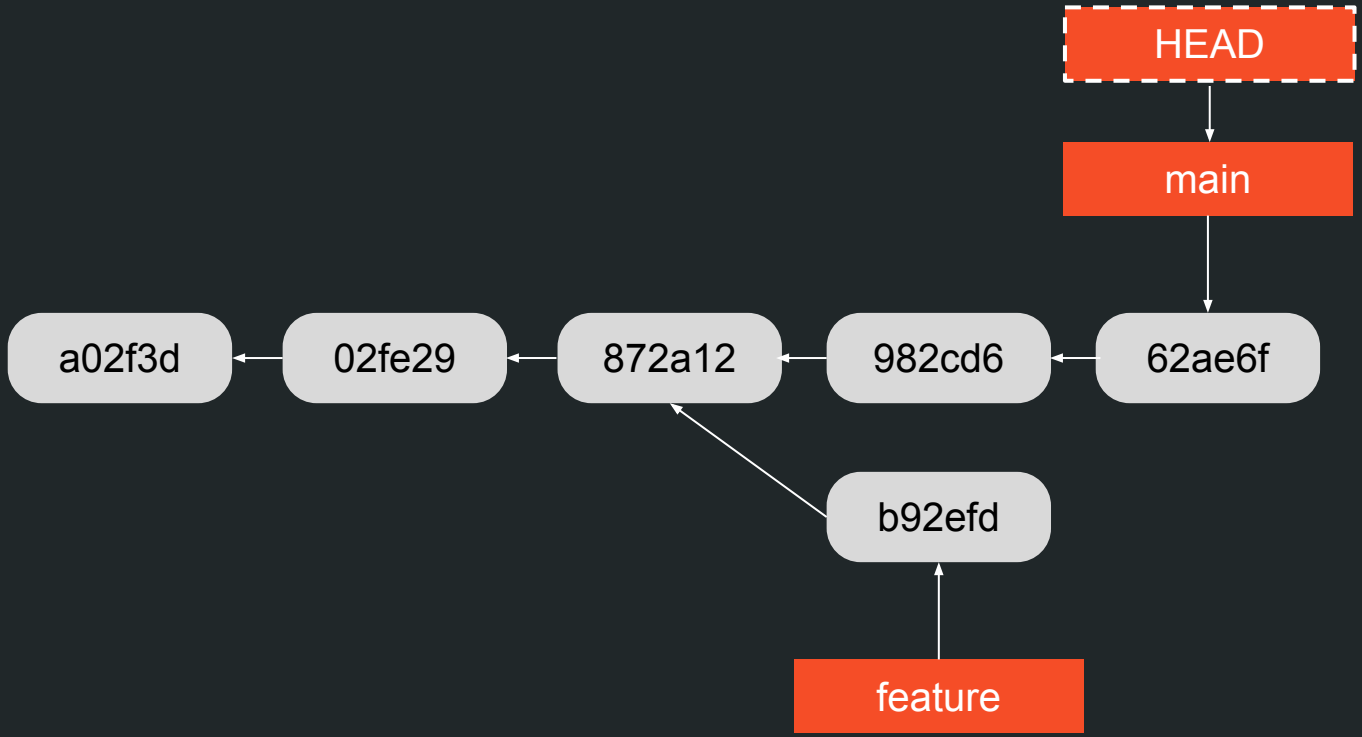

$ cat file
a
--- a/file
+++ b/file
@@ -1 +1,2 @@
a
+b
--- a/file
+++ b/file
@@ -1 +1,2 @@
a
-b
$ cat file
a
b
a02f3d
02fe29
872a12
982cd6
62ae6f
73ba2d
b92efd
--- a/file
+++ b/file
@@ -1 +1,2 @@
a
+b

Rebase: rewriting the past



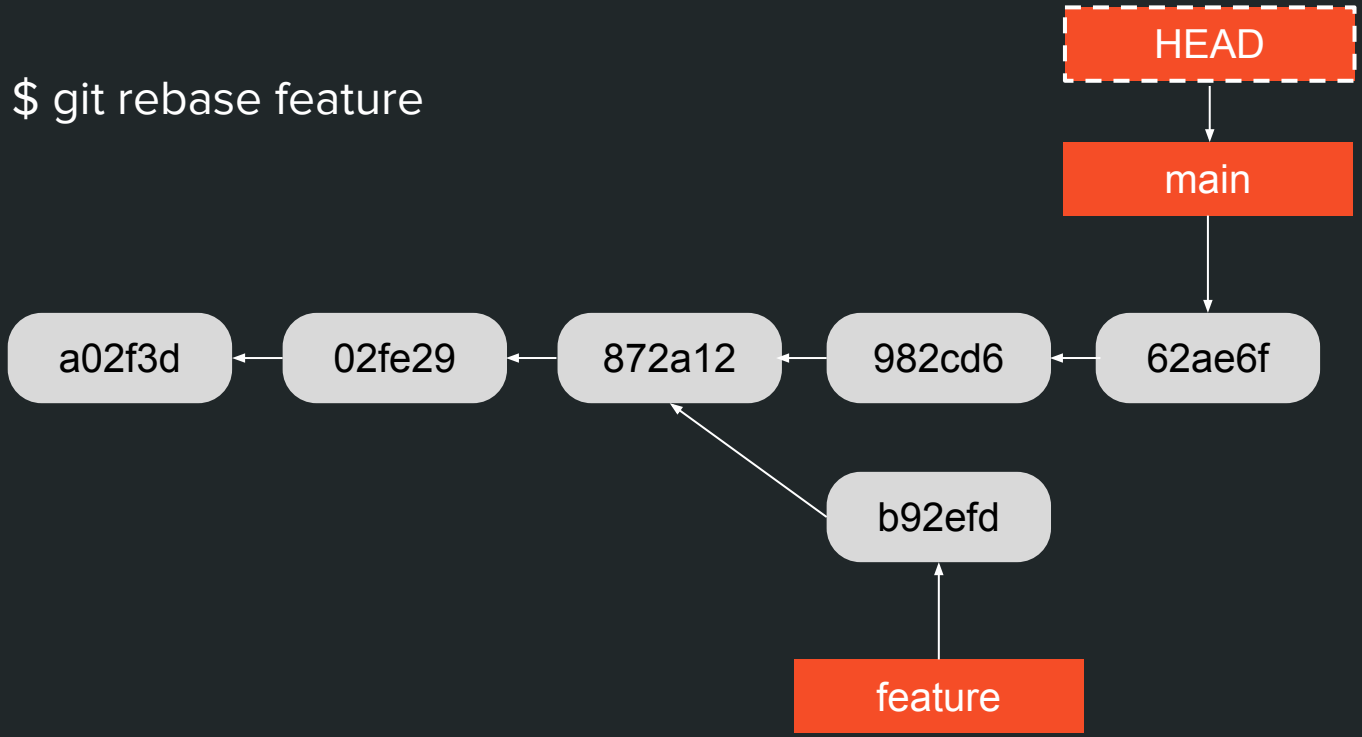
https://memegenerator.net

Rebase


HEAD
main
a02f3d
02fe29
872a12
982cd6
62ae6f
b92efd
feature

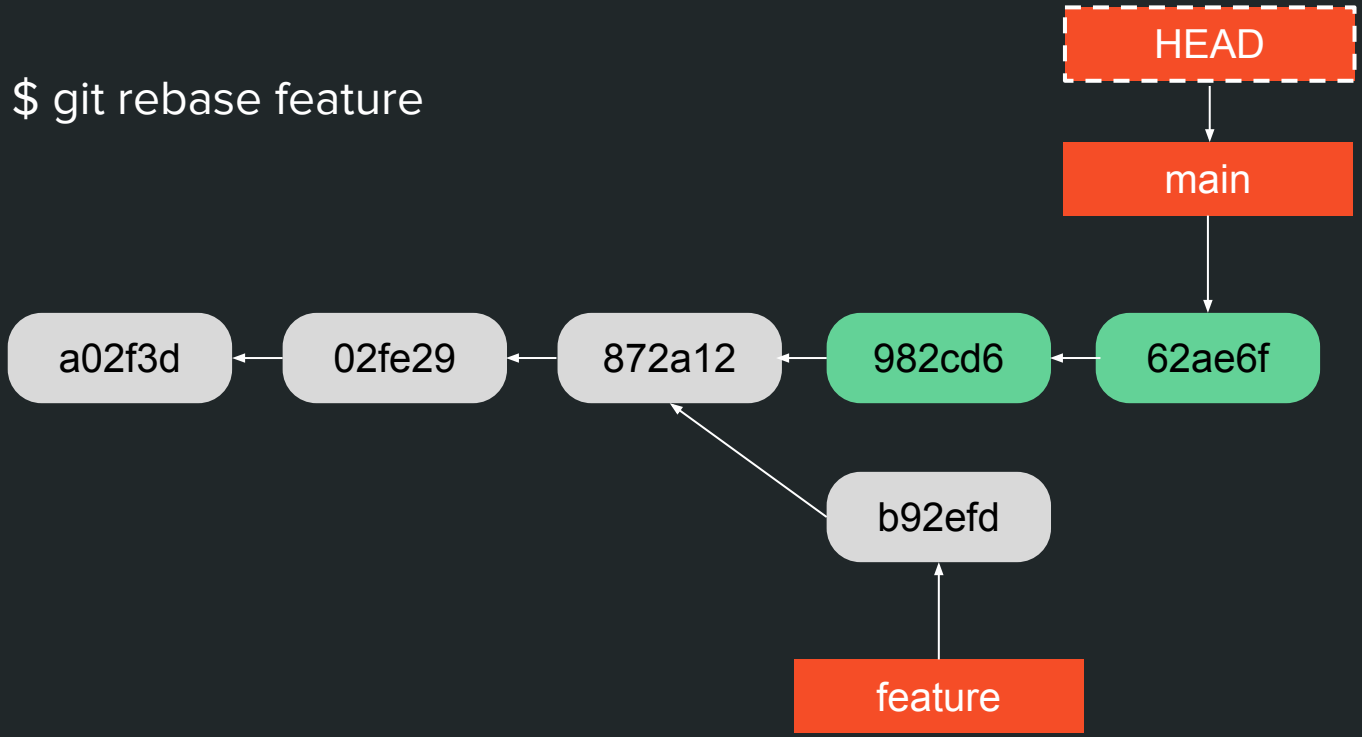
Rebase

$ git rebase feature



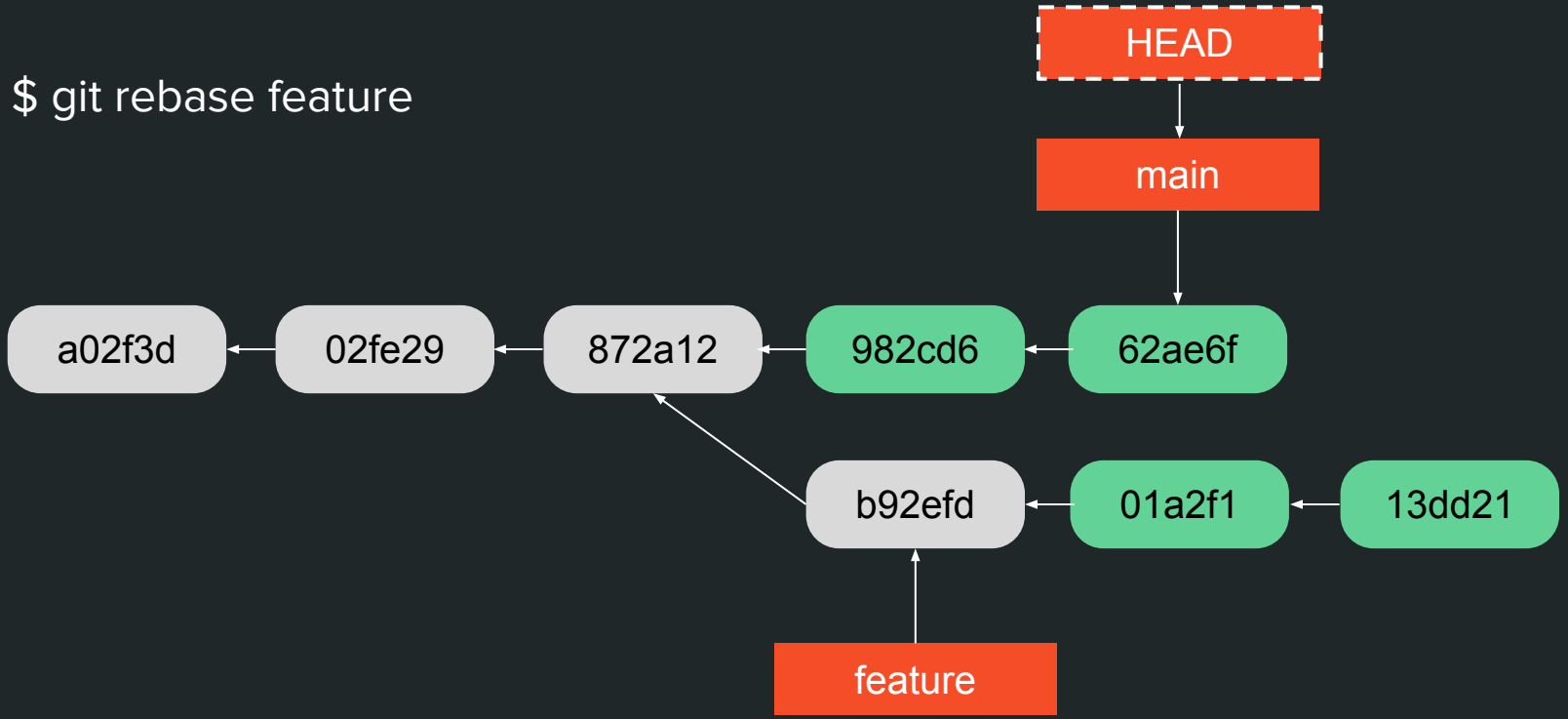
Rebase

$ git rebase feature



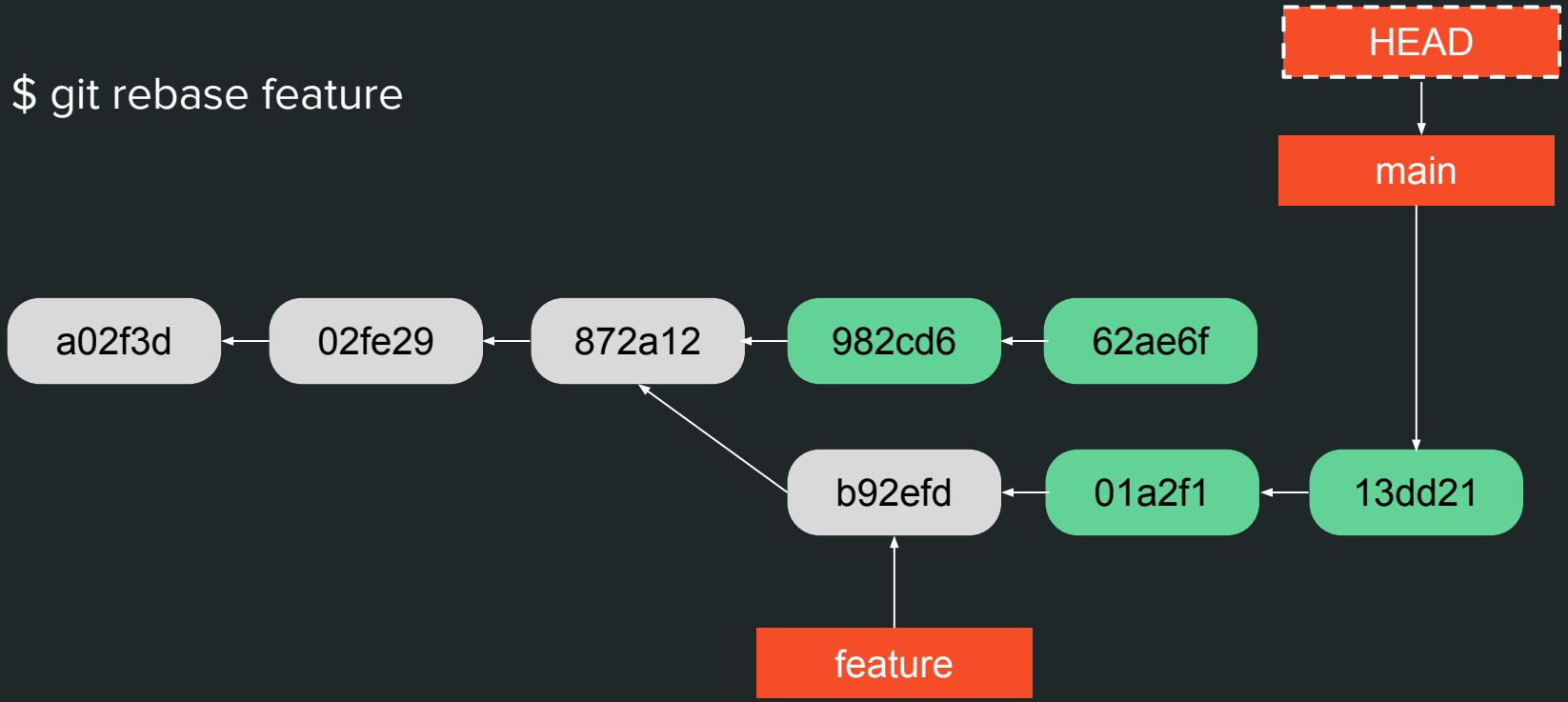
Rebase

$ git rebase feature



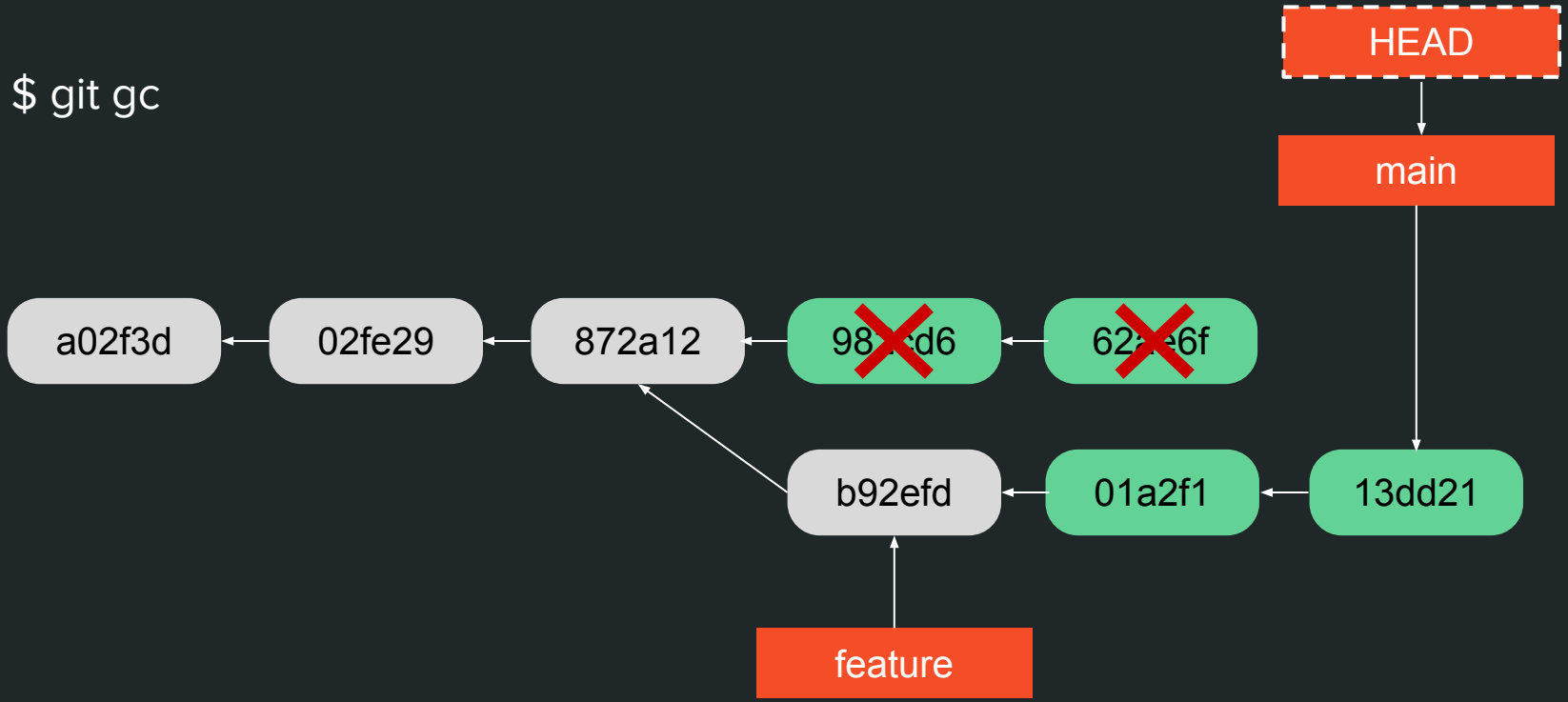
Rebase

$ git rebase feature



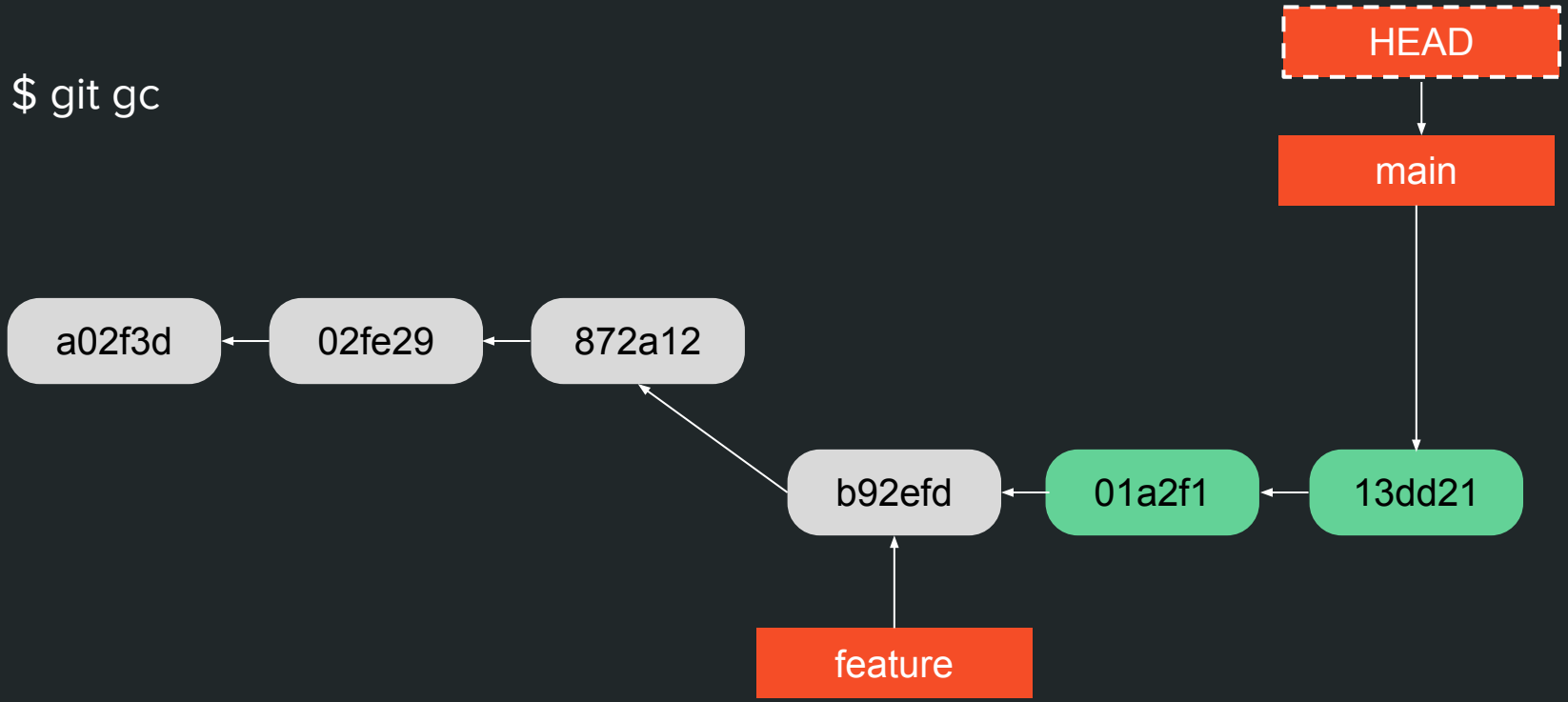
Rebase

$ git gc



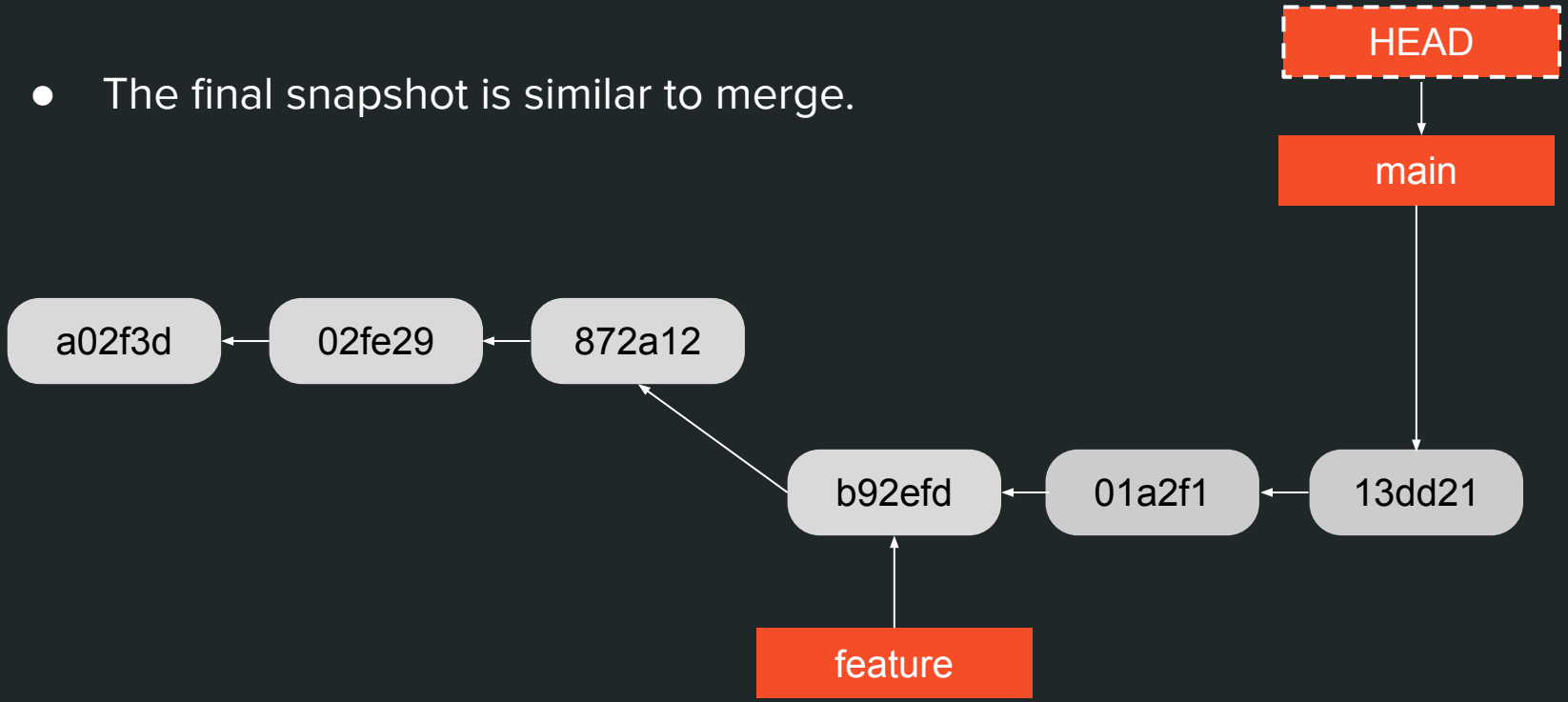
Rebase

$ git gc



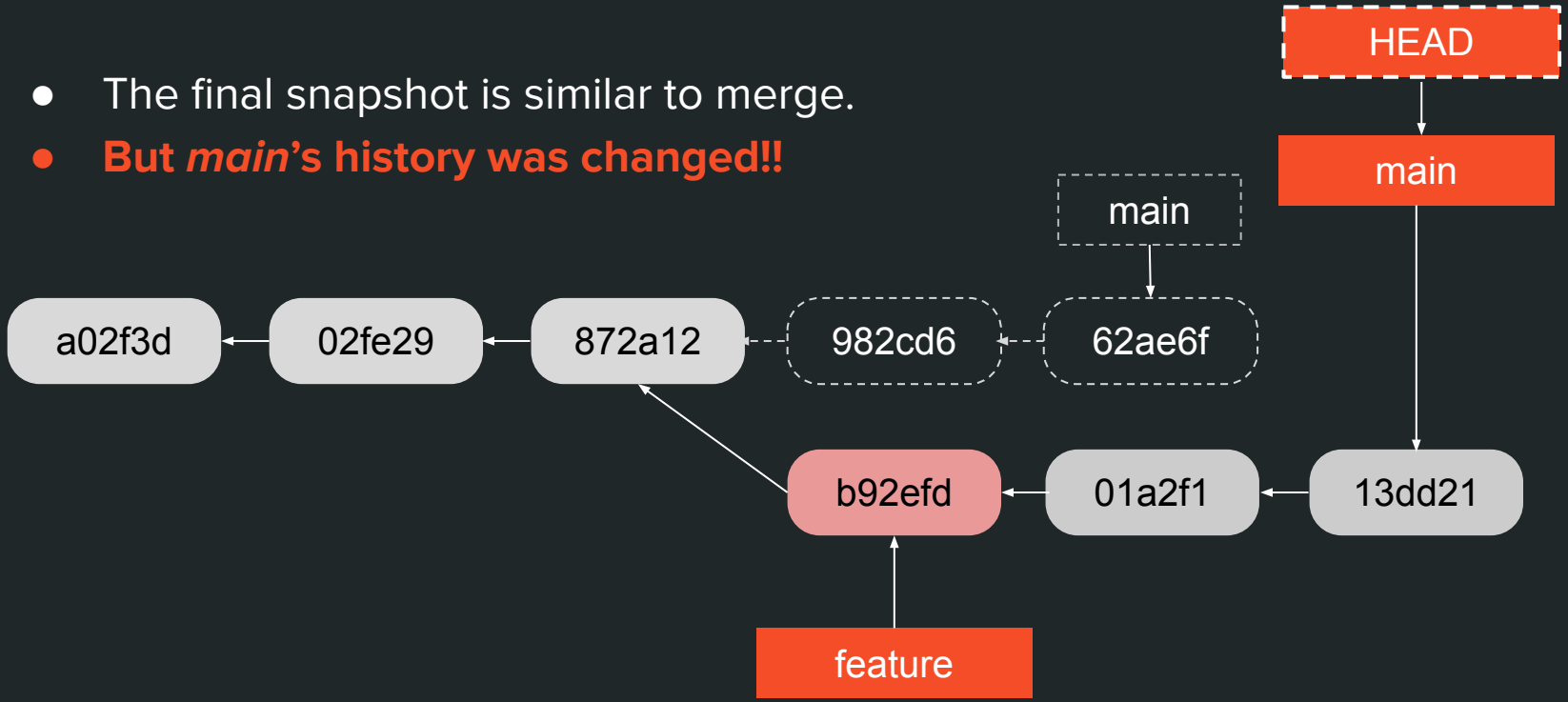
Rebase

- The final snapshot is similar to merge.



Rebase

- The final snapshot is similar to merge.
- **But *main*'s history was changed!!**



Rebase’s rule of thumb

- **Good use case:** organizing a personal branch before merging to main.

- **Possibly troublesome:** rewriting a public branch, being used by others.

(that others depend on)

http://jdduarte.com/git-training/#/

$ git rebase -i

- Add
- Remove
- Reorder
- Edit/amend
- Join multiple commits
- Run cmd for each commit
- ...

Thanks

Extra Git Tips

- *$ git help glossary*
- *$ git help revisions*
- *$ git reflog*
- *$ git blame / git log -S*
- *$ git bisect*
- *$ git worktree*
- *$ git clone --depth=1 [--filter=blob:none]*

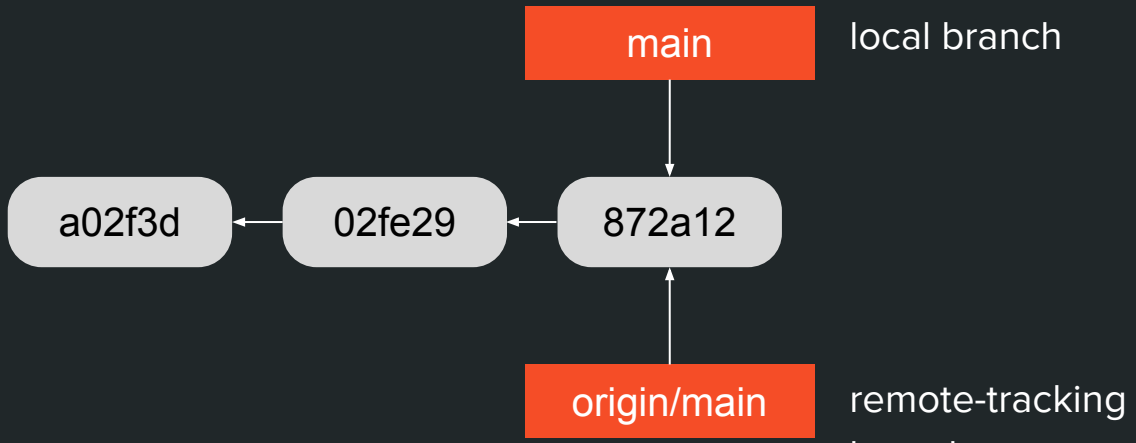
References

1. **The Zen of Git**, Tianyu Pu: https://speakerdeck.com/tianyupu/the-zen-of-git
2. **Pro Git**, Scott Chacon and Ben Straub: https://git-scm.com/book/en/v2
3. **Git Docs:** https://git-scm.com/docs/
4. **Git For Computer Scientists**, Tommi Virtanen: https://eagain.net/articles/git-for-computer-scientists/

Extra: Remotes

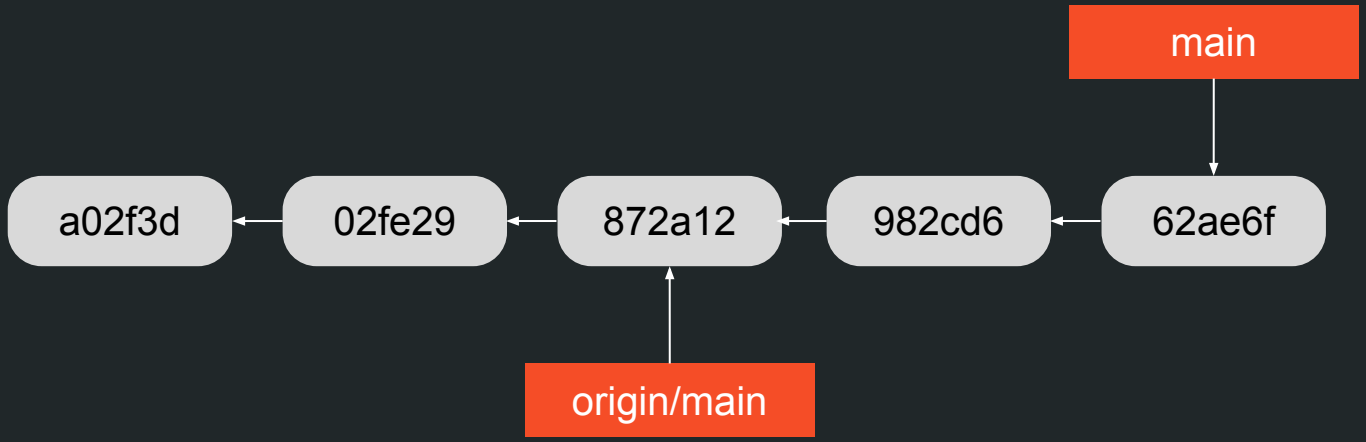
Remote-tracking Branches

$ git clone



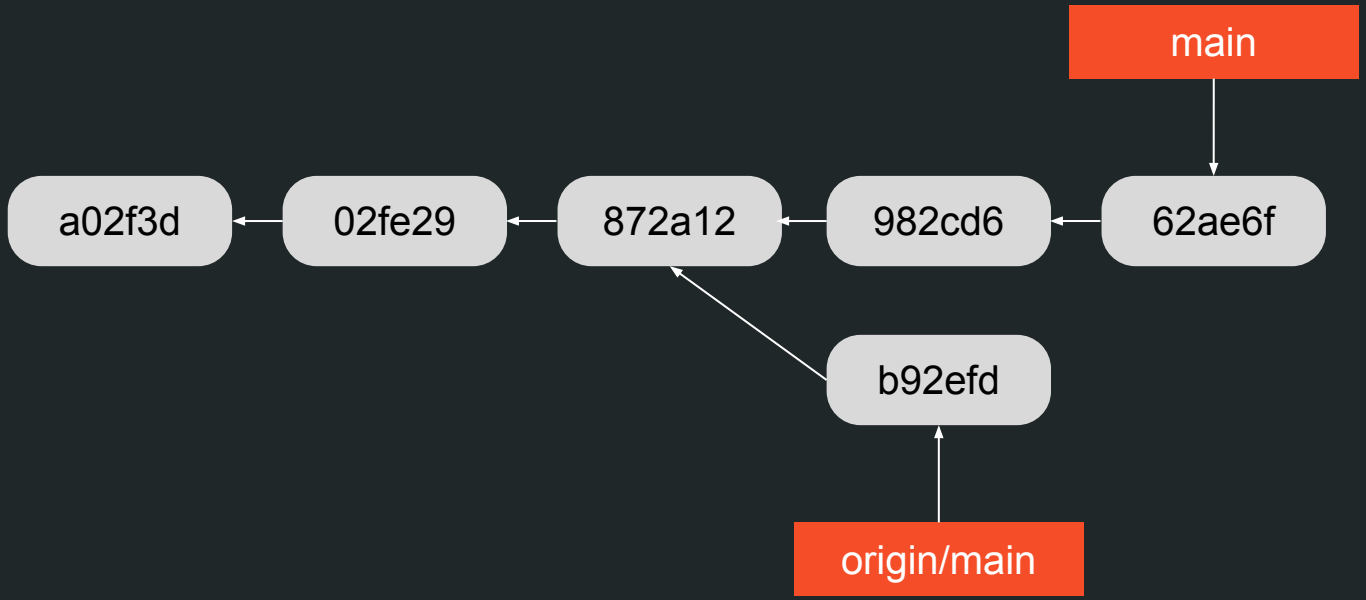
Remote-tracking Branches

$ git commit (2x)



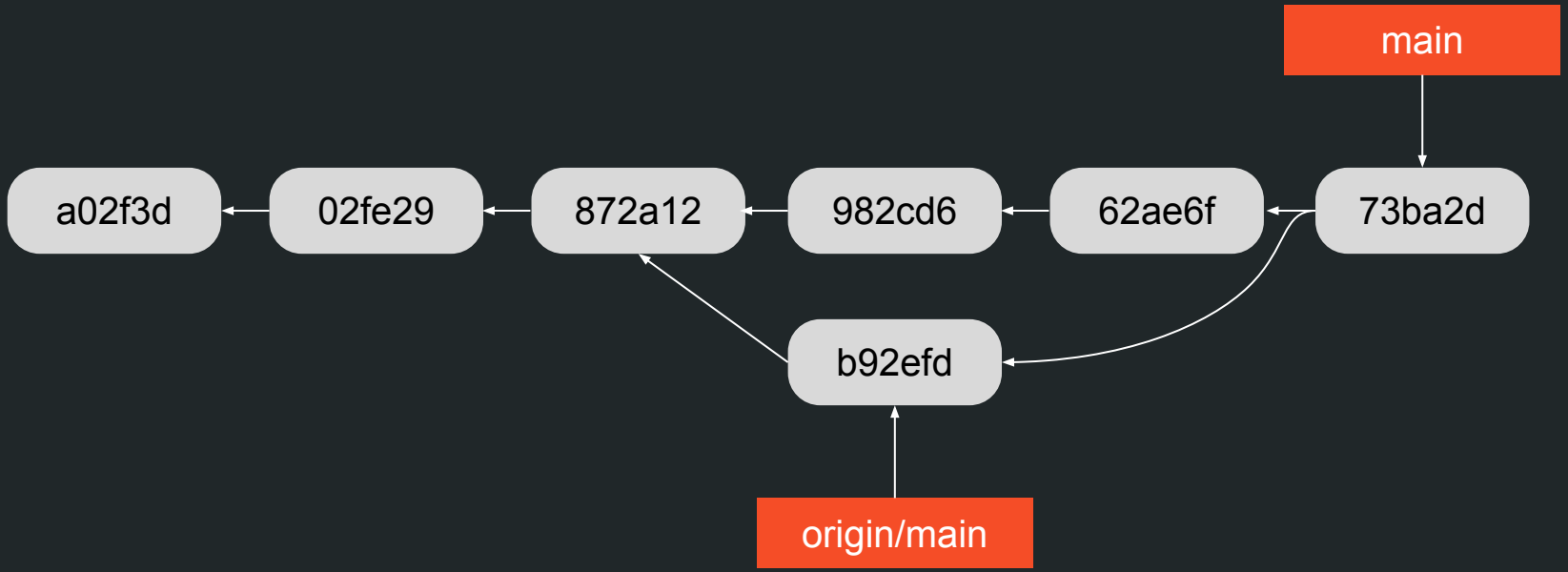
Remote-tracking Branches

$ git fetch origin



Remote-tracking Branches

$ git merge origin/main



Remote-tracking Branches

- We just did a manual “git pull”